PROGRAMACION

BASIC

MSX

S-T

BASIC

MSX

PROGRAMACION

BASIC

MSX

Enseñanza autodidacta
primer nivel



PROGRAMACION

BASIC

MSX

Enseñanza autodidacta
primer nivel

SONY

LOGIC CONTROL®

La informática está de moda.
Dentro de la informática, los lenguajes que nos permiten dirigir la potencia del ordenador a la obtención de un determinado resultado, centran el máximo nivel de interés.
El BASIC *es el más popular de estos lenguajes.*

En las páginas siguientes se resumen nuestros esfuerzos por hacer accesible a los ambientes no informáticos, el MSX-BASIC, *en forma autodidacta.*

Al igual que el castellano tiene pequeñas diferencias según el lugar donde se practique (España, sudamérica, etc), también el "idioma" BASIC *difiere en alguna de sus características según sea el computador en el que quiera ejecutarse. He aquí lo importante que es el* "MSX" *que precede a la palabra* "Basic" *en nuestra portada.*
"MSX" *es el primer estandard que aglutina a una gran mayoría de los computadores domésticos del mercado. Ello quiere decir que este manual puede ser aplicado sobre cualquier computador, suministrado por cualquier proveedor, que responda al estandard* "MSX".

Al igual que en una buena aventura, en el aprendizaje del BASIC *ha de haber momentos excitantes, situaciones de resultado incierto, y un final feliz. También el desánimo puede hacer su aparición, pero la constancia hará que el mismo sea superado.*

Aquí, el tiempo no cuenta. No se trata de competir con nadie. El mundo ignoto del computador esperará pacientemente a que cada alumno, a la medida de su interés y sus posibilidades, vaya conociendo y dominando las diversas facetas del mismo.

Después de esta primera parte de nuestra aventura, comprobarás que un computador es como un paisaje por el cual se puede pasear en varias dimensiones y profundidades.

Si, una vez finalizado el presente curso, el alumno ha conseguido establecer un gratificante diálogo con el computador, y siente despertar nuevas inquietudes más allá de los límites de los conocimientos ya adquiridos, nuestro objetivo se habrá conseguido. Entonces, un nuevo curso estará a su disposición para seguir andando juntos en este apasionante camino que la constante evolución de la electrónica pone a nuestro alcance.

No hay limitaciones previas para la utilización de este manual. Ni la edad, ni la profesión, ni la condición intelectual.

El interés, la constancia y la ilusión son las únicas condiciones previas que nuestro lector debe poseer.

LOGIC CONTROL
Abril 1985

INDICE TEMATICO POR SESIONES

SIGNIFICADO DE LOS SIMBOLOS UTILIZADOS EN EL SEGUIMIENTO PRACTICO DE LOS EJERCICIOS

Cuando veas una mano  *significa que se ha de teclear (apretar una ó varias teclas del teclado) lo que venga a continuación, que pueden ser dos cosas:*

- UN TEXTO: *Para pedirte que escribas* "Hola", *se te indicará:*

 "Hola"

- LA REPRESENTACION DE UNA TECLA CONCRETA. *Para pedirte que aprietes una tecla concreta, se te indicará el nombre o figura de dicha tecla, envuelta en una línea. Por ejemplo, para que teclees la tecla* RETURN, *se te indicará:*

 (RETURN)

Cuando haya que apretar simultaneamente dos teclas, se indica mediante un símbolo + . *Por ejemplo, si queremos que aprietes la tecla* SHIFT *y mientras la mantienes apretada, teclees la tecla* HOME, *se te indicará:*

 (SHIFT) + (HOME)

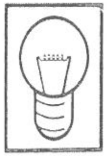
Hay teclas que tienen dibujados dos símbolos diferentes, uno superior y otro inferior. Para las mayúsculas (en el caso de una tecla que sólo tenga una letra), y los símbolos que están en la parte superior de las teclas dobles, hay que apretar simultaneamente (a la vez) la tecla SHIFT *y la tecla correspondiente. Hay dos teclas* SHIFT *en el teclado, una a la izquierda y otra a la derecha. Por ejemplo:*

 "Hola"

se escribiría así:

 (SHIFT) + (") (SHIFT) + (H)

(o) (l) (a) (SHIFT) + (")



IL

Este símbolo señala los puntos curiosos e importantes de cada sesión. Humorísticamente los llamaremos Ideas Luminosas (IL) y vendrán numeradas por si en alguna otra parte del libro nos referimos a alguna IL en especial.

Ø

Este símbolo representa el número cero. Se cruza una línea encima de él para no confundirlo con la cuarta vocal, la "o". Es decir, siempre que escribamos un cero, éste estará atravesado por una raya diagonal.

En el lenguaje BASIC *hay una serie de palabras que más adelante conocerás perfectamente. A continuación reflejamos algunas de ellas. No te molestes en aprenderlas de memoria o comprender exactamente su significado. Solo léelas atentamente y acuérdate que están aquí para consultarlas cuando las encuentres en el libro y quieras ver su significado.*

Como muchas palabras BASIC *proceden del inglés, te ponemos su significado en inglés y su traducción al castellano. Esta traducción no es muy literal, sinó mas bien explicativa.*

MINIVOCABULARIO BASIC MSX

Basic	Inglés	Castellano
ABS	ABSOLUTE	Valor absoluto
ASC	AMERICAN STANDARD CODE	Código Estandard Americano
AUTO	AUTOMATIC	Automático
BEEP	BEEP	Pitido
BIN	BINARY	Binario
CHR	CHARACTER	Carácter
CLEAR	CLEAR	Limpiar
CLS	CLEAR SCREEN	Limpiar Pantalla
CONT	CONTINUE	Continuar
CSRLIN	CURSOR LINE	Línea del cursor
DATA	DATA	Datos
DELETE	DELETE	Borrar
DIM	DIMENSION	Dimensión
END	END	Fin
ERASE	ERASE	Borrar
FOR	FOR	Para
FREE	FREE	Libre
GOSUB	GO SUBROUTINE	Ir a la subrutina
GOTO	GO TO	Ir a
HOME	HOME	Origen
IF	IF	Si...
INPUT	INPUT	Introducir
INKEY	IN KEY	Tecla introducida
INSTR	IN STRING	Dentro de la cadena
INT	INTEGER	Entero
KEY ON	KEY ON	Teclas dispuestas
KEY OFF	KEY OFF	Teclas no dispuestas
LEFT	LEFT	Izquierda
LEN	LENGHT	Longitud
LINE INPUT	LINE INPUT	Línea introducida

Basic	Inglés	Castellano
LIST	LIST	Listar
LOCATE	LOCATE	Situar
MID	MIDDLE	Del medio
NEXT	NEXT	Siguiente
NEW	NEW	Otro, nuevo
ON...GOSUB	ON...GO TO SUBROUTINE	Según...ir a subrutina
ON...GOTO	ON...GO TO	Según...ir a
POS	POSITION	Posición
PRINT	PRINT	Escribir
PRINT USING	PRINT USING	Escribir usando
READ	READ	Leer
REM	REMARK	Comentario
RENUM	RENUMBER	Volver a numerar
RESTORE	RESTORE	Restablecer
RETURN	RETURN	Volver a
RIGTH	RIGTH	A la derecha
RND	RANDOM	Al azar
RUN	RUN	Correr, ejecutar
SHIFT	SHIFT	Cambiar por el de arriba
SGN	SIGN	Signo
SPACE	SPACE	Espacio en blanco
SPC	SPACE	Espacio en blanco
STOP	STOP	Parar
STR	STRING	Cadena de caracteres
STRING	STRING	Cadena de caracteres
SWAP	SWAP	Intercambiar
TAB	TABULATE	Encolumnar
TIME	TIME	Hora
VAL	VALUE	Valorar
WIDTH	WIDTH	Ancho

PARA EMPEZAR A TRABAJAR

(Solo si el computador aún no ha sido instalado)

- *Poner el* MSX-COMPUTER *sobre una superficie plana.*
- *Situarse próximos a un receptor de TV dotado de UHF. (Segundo canal)*
- *Conectar el* MSX-C *con la TV mediante el cable apropiado, suministrado con el computador, uniendo la salida trasera RF con la toma de antena del TV.*
- *Enchufar el* MSX-C *y la TV. Encender ambos. El* MSX-C *tiene su interruptor en el lateral.*
- *Sintonizar la TV en las cercanias del canal 36 del UHF.*
- *Poner el volumen de la TV a cero.*
- *Sintonizar finamente hasta ver nítida la siguiente pantalla:*



- *Apagar el* MSX-C

En este momento se observará que el MSX-C *ha abandonado el control de la pantalla. Es evidente que el* MSX-C *se encarga de la pantalla y por*

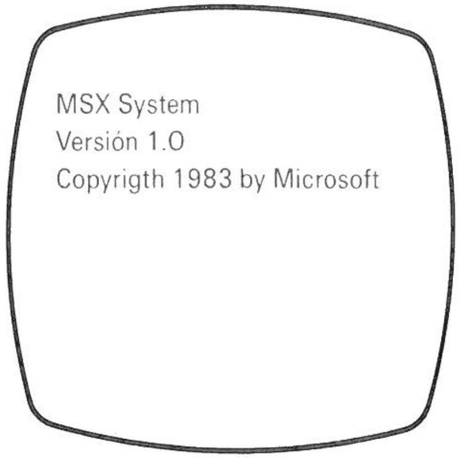
ella emite información. La pantalla es un periférico de salida de información.

La palabra PERIFERICO *tiene origen griego y significa* "lo que está alrededor".

Pantallas, mandos para juegos, impresoras, etc. Todo esto que está "alrededor" *del* MSX-COMPUTER, *son periféricos.*

ARRANQUE

- *Encender el* MSX-C, *mediante el interruptor situado en el lateral izquierdo. Al cabo de 2 segundos se verá la siguiente pantalla:*



Explicación: (solo por curiosidad)

MSX-System:
Nombre del sistema operativo del MSX-C.
MSX *viene de MicroSoft Extended.*

Versión 1.0:
Versión original del sistema operativo.
1 = Primera edición
.0 = Sin modificaciones

Copyrigth 1983 by Microsoft:
Derechos de autor registrados por Microsoft en 1983.

- *A los cuatro segundos aparecerá esta otra pantalla:*



Explicación:

MSX BASIC: *Basic del sistema* MSX.

Versión 1.0: *Versión original sin modificaciones.*

Copyrigth....: *Derechos de autor registrados.*

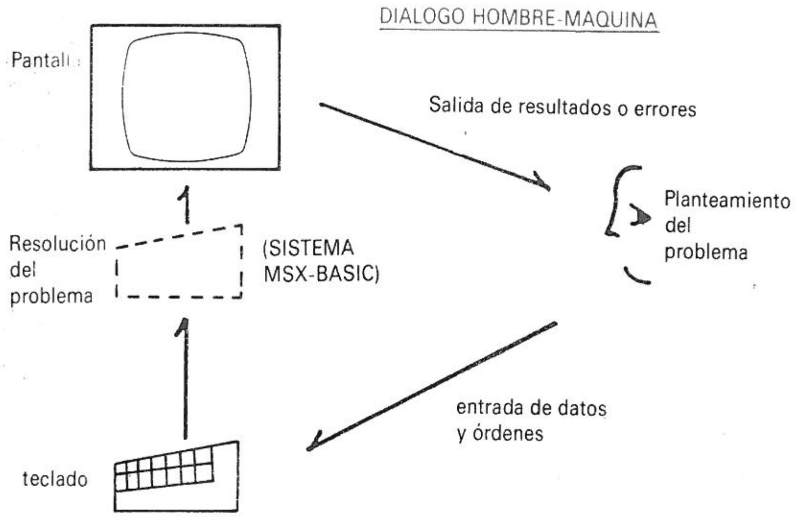
28815 Bytes free: *número de posiciones (octetos) que quedan libres para los datos y programas. Un octeto es una porción de memoria en la que cabe una letra o dos números.*

OK: *Quiere decir que un programa ha terminado su trabajo y el* MSX-C *queda a la espera de nuevas órdenes o datos.*

□ : *Cursor o señal en forma de cuadrado, que señala el lugar donde va a escribirse el primer dato o información que tecleemos. Cada vez que entramos una letra o un número, el cursor indica la próxima posición para entrar una letra o número mediante un desplazamiento.*

COLOR AUTO GOTO LIST RUN: *Comandos y sentencias de Basic que no son necesarias teclear letra a letra pues, por su frecuencia de uso, están automatizadas sobre las cinco teclas superiores de color gris llamadas teclas de función.*

Ya nos podemos comunicar con el MSX-C *a través de su "entrada", el teclado, y su "salida", la pantalla.*



SESION 1

Una SESION *es el diálogo que tenemos con el* MSX-C *desde que lo encendemos hasta que lo apagamos.*

En esta primera sesión aprenderemos a mover el cursor por la pantalla y otras cosas muy interesantes.

ENCIENDE EL MSX-C

Después de encender el MSX-C, *la pantalla de TV expone los dos textos que hemos visto anteriormente. Fijémonos en la última línea, en la parte baja de la pantalla, donde se ven las palabras:*

COLOR AUTO GOTO LIST RUN

Observarás que son cinco, tantas como teclas grises hay en la parte superior del teclado.

Comprobemos que esas teclas se corresponden con las palabras de la última línea.

F1 *(aprieta* **F1***) y se escribe la palabra* COLOR

F2 *y se escribre la palabra* AUTO

F3 *y se escribe la palabra* GOTO

F4 *y se escribe la palabra* LIST

F5 *y se escribe la palabra* RUN...

y además se escribe:

SYNTAX ERROR *(indicación de que hemos hecho algo mal)*

OK

Esto sucede porque RUN *es la orden de ejecutar el programa que esté en la memoria del* MSX-C *pero la sentencia* RUN *ha de ir sola en una línea, de lo contrario da error. Comprobémoslo:*

(F1) (F2) (F3) (F4)

(⇩)

(F5)

Y verás que no da error pues RUN *ha funcionado correctamente, aunque no hace nada pues no hay ningún programa en memoria. De todas maneras lo intenta, no encuentra nada y termina su tarea, tal como nos lo indica el* OK *y el cursor.*

Las palabras

COLOR AUTO GOTO LIST RUN

que figuran en la última línea de la pantalla son un recordario de la función que tienen las cinco teclas superiores, las cuales se llaman teclas de función, porque tienen una función asignada. Como de momento no las vamos a usar, suprimiremos esta última línea.

KEY OFF (RETURN)

"KEY OFF" *equivale al castellano* "Teclas no dispuestas".

Efectivamente, la línea

COLOR AUTO GOTO LIST RUN

ha desaparecido.

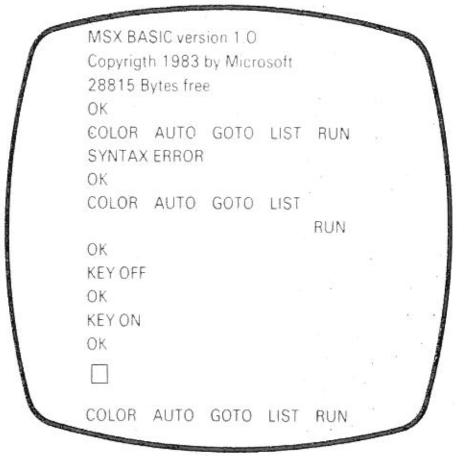
KEY ON (RETURN)

y verás como aparece otra vez la línea orientativa. "KEY ON" *equivale al castellano* "teclas disponibles".

Si te equivocas escribiendo, retrocede con (⇦) *hasta la equivocación, y repite desde ahí.*

Si ya has apretado (RETURN) *y el ordenador da mensaje de error, repite toda la línea como si no hubieras hecho nada.*

En este momento debes tener a la vista una pantalla como la que se describe a continuación. (Si ha habido errores, puedes tener alguna línea más).



Con la tecla (⇧) *situa el cursor sobre la primera letra de la linea que dice:*

KEY OFF

Si te pasas de línea hacia arriba haz bajar el cursor con la tecla

(⇩)

Ahora, fíjate lo que va a pasar si sobre esta posición haces

(RETURN)

¿Qué ha pasado al apretar RETURN?

Pues que la instrucción KEY OFF ha sido obedecida y la última línea (COLOR.......) ha sido borrada.

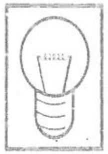
Baja ahora el cursor, si no está ya ahí, hasta la línea que dice:

KEY ON

y ejecuta:

(RETURN)

Comprueba que la última línea ha vuelto a aparecer.



IL1

> Cada vez que se apriete RETURN, esté donde esté el cursor, el MSX-C leerá la línea completa donde está el cursor e intentará hacer algo con ella.

Si lo que encuentra es correcto, lo ejecutará o lo introducirá en memoria. En cualquier otro caso comunicará un error y quedará a la espera.

Antes de terminar esta sesión, aprenderemos a limpiar la pantalla. Utilizaremos la sentencia CLS (limpiar pantalla) que va a borrar toda la pantalla menos la última línea. Para borrar también esta última línea, podemos utilizar también KEY OFF, como ya sabes.

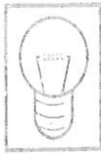
Vamos a utilizar las dos instrucciones a la vez:

KEY OFF: CLS (RETURN)

Al ejecutarse esta línea vemos que han pasado dos cosas, aunque tan rapidamente, que parecen una sola.

1ª.- Se ha borrado la línea inferior

2ª.- Se ha borrado todo lo demás que estaba en la pantalla.



IL2

> En una misma línea, SEPARADAS POR DOS PUNTOS, pueden ir varias órdenes o instrucciones.

Observarás que hay todavía un OK en la pantalla.

Para borrar totalmente la pantalla, una vez usado KEY OFF para borrar la línea inferior:

(SHIFT) + (HOME)

Cada vez que se teclea lo anterior, esté donde esté el cursor, la pantalla se borra TOTALMENTE y el cursor se situa en el origen.

Desplaza el cursor hasta el centro de la pantalla

(⇩) + (⇨)

Escribe ahora A B C

y ahora

(HOME)

Comprueba que el cursor se ha desplazado al origen pero la información de la pantalla queda intacta.

Si queremos que también se borre todo lo de la pantalla, recuerda:

(SHIFT) + (HOME)

APAGA EL MSX-C

FIN DE LA 1ª SESION

IMPORTANTE: *Intenta repetir la sesión de memoria. Experimenta acerca de lo que has aprendido.*

¿Que hemos aprendido?

- *Conectar el* MSX-COMPUTER
- *Suprimir o representar la línea de teclas de función.*
- *Comprender la importancia de la posición del cursor.*
- *Mover el cursor por la pantalla.*
- *Las sentencias* KEY OFF, KEY ON y CLS.
- *La posibilidad de varias sentencias por línea a condición de separarlas por dos puntos.*

EJERCICIOS SESION 1

Sin consultar otras páginas de este manual, ejecuta los siguientes ejercicios.

1°.- *Apaga y enciende de nuevo a continuación el* MSX-C

2°.- *Suprime la línea* "COLOR AUTO..."

3°.- *Tocando una sola vez una sola tecla, situar el cursor al inicio de la pantalla, sin borrar el contenido de la misma.*

4°.- *Haz aparecer la línea* "COLOR AUTO..."

5°.- *Utilizando dos instrucciones en una misma línea, borra la pantalla y la línea inferior* "COLOR AUTO..."

6°.- *Haz que en la pantalla quede unicamente el cursor.*

NOTA: *Si no hallas por ti mismo la solución, mira la solución en la página siguiente y repasa esta sesión.*

SOLUCION A LOS EJERCICIOS DE LA SESION 1

1°.- *Repasar inicio sesión 1*

2°.- KEY OFF (RETURN)

3°.- (HOME)

4°.- KEY ON (RETURN)

5°.- KEY OFF: CLS (RETURN)

6°.- (SHIFT) + (HOME)

SESION 2

En la sesión anterior hemos aprendido, entre otras cosas, a borrar la pantalla. En esta sesión vamos a aprender a escribir en ella algunas cosas sencillas tales como números, letras, palabras y frases.

Comencemos por encender el MSX-C *y:*

KEY OFF (RETURN)

y se ha borrado la última línea.

(SHIFT) + (HOME)

Se ha borrado el resto de la pantalla.

PRINT 9 (RETURN)

Como ves, le estamos ordenando al MSX-C *que escriba el número 9 y lo ha hecho en la siguiente línea, dejando un espacio en blanco a su izquierda, antes de escribirlo.*

Lo escribe en la siguiente línea porque en la que estaba era donde ha leido la orden de escribir.

Deja un espacio en blanco porque este espacio se reserva para poner el signo cuando éste es negativo. Si es positivo, como en nuestro ejemplo, no pone el signo.

En aritmética, se hace lo mismo, los positivos no suelen llevar escrito el signo.

PRINT-9 (RETURN)

Ahora vemos que escribe el número con el signo –.

El MSX-C *deja siempre una posición para el signo antes de imprimir los números. No pone nada si el número es positivo y pone un – si es negativo.*

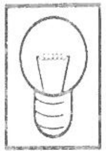
PRINT 27/3 (RETURN)

Observa que la sentencia PRINT puede escribir el resultado de los cálculos que tenga a continuación.

En este caso era una división, ya que el signo de la división es la barra inclinada. El resultado era, como ves, 9.

PRINT "27/3 (RETURN)

¿Que ha pasado?. Que el cálculo de la división no ha sido realizado.



IL3

> Si ponemos unas comillas después de PRINT, lo que hay después de las comillas se imprime tal cual está.

Por eso ahora no imprime el resultado de la división, si no lo que hay después de las comillas.

Cuando hay comillas después del PRINT esta sentencia no distingue los cálculos y toma los números como cualquier otro caracter o símbolo y los imprime tal como están. Por eso lo hace en la primera posición de la línea, pues no tiene que preveer ningún signo.

PRINT "27/3 PEPE (RETURN)

Nuevamente lo toma todo como una línea o cadena de caracteres, y no efectúa el cálculo de la división.

Cambiemos la posición de las comillas.

PRINT 27/3" PEPE (RETURN)

Observa como ahora sí ha realizado el cálculo de la división y ha tomado lo que hay después de las comillas como una cadena de caracteres. Ahora prueba:

PRINT 27/3" PEPE 8 + 3 (RETURN)

El segundo cálculo no lo realiza, como ya sabes, porque está a continuación de las comillas.

Prueba ahora:

PRINT 27/3" PEPE" 8 + 3 (RETURN)

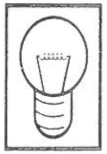
Observa que ahora si ha realizado los cálculos correctamente.

Como ves, se pueden realizar cálculos también después de unas comillas. Basta con cerrar las comillas antes del nuevo cálculo.

Las comillas se pueden abrir y cerrar tantas veces como queramos que el MSX-C escriba cadenas de caracteres, textos y mensajes.

Observa que el resultado de 8 + 3 = 11, tiene un espacio delante para el signo, pero éste no sale porqué es positivo.

Todo lo que está entre comillas se deja tal y como está, no se opera ni varia interiormente. Se puede decir que es constante.



IL4

> Constante de cadena es el conjunto de información que va entre comillas. Puede estar compuesta de números, letras, símbolos y espacios en blanco.

Vamos a ordenar al MSX-C que emita un mensaje. Las instrucciones y ordenes al ordenador, se pueden escribir en minúsculas o mayúsculas, a tu comodidad.

PRINT "SOY UN COMPUTADOR MSX" (RETURN)

Te falta poco para que domines la técnica de que el MSX-C emita mensajes por la pantalla.

APAGA EL MSX-C

FIN DE LA SESION 2

¿Que hemos aprendido?

- *Imprimir un número, positivo o negativo.*
- *Que los números van precedidos de la información sobre el signo y seguidos de un espacio en blanco para separarlos de otras informaciones.*
- *Que* PRINT *imprime resultados de cálculos.*
- *Como influyen las comillas en el control de impresión.*
- *Qué es una cadena de caracteres.*
- *Escribir un mensaje en la pantalla.*

EJERCICIOS SESION 2

Sin consultar otras páginas de este manual, ejecuta los siguientes ejercicios.

7°.-*Que el* MSX-C *escriba* -385.

8°.-*Que el* MSX-C *escriba el resultado de la división* 36 *entre* 9. *(Es decir, 4)*

9°.-*Que el* MSX-C *escriba* 3 + 4 PEPE.

10°.-*Que el* MSX-C *escriba el resultado de* 50 + 35, *a continuación* PAMPLONA, *y el resultado de* 25 + 5.

NOTA: *Si no hallas por ti mismo las soluciones, búscalas en la próxima página.*

SOLUCION A LOS EJERCICIOS DE LA SESION 2

7º.- PRINT -385 (RETURN)

8º.- PRINT 36/9 (RETURN)

9º.- PRINT "3 + 4 PEPE (RETURN)

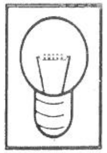
10º.- PRINT 50 + 35 "PAMPLONA" 25 + 5 (RETURN)

SESION 3

El MSX-C *tiene* MEMORIA. *Tú ya sabes que la memoria sirve para recordar, para almacenar recuerdos. Lo que un dia vimos, oimos, sentimos, a veces está presente en nuestra memoria. Alguna vez algo nos* "entró" *en la memoria y puede que lo recordemos.*

La memoria del MSX-C, *si lo usamos bien, no solo* "puede" *recordar, sinó que* RECUERDA SIEMPRE *lo que le hemos* "entrado" *a condición de que sigamos unas pequeñas reglas.*

La primera regla es sencilla:



IL5

> *Todo lo que introducimos en la memoria del* MSX-C, *si no lo cambiamos o borramos, está presente en dicha memoria y puede usarse mientras no apaguemos el* MSX-C.

¿Qué podemos introducir en la memoria del MSX-C?

en la memoria del MSX-C *podemos introducir datos.*

Suponte que un amigo te dice, señalando un grupo de gente:

- "Aquel de la barba se llama Juan".

En tu memoria queda una pareja de ideas que, muy sencillamente expresada, es así:

Barba - Juan

Si tu memoria es buena, cada vez que veas aquella misma barba en aquella misma cara, te acordarás del nombre "Juan".

Haz:

KEY OFF (RETURN)

(SHIFT) + (HOME)

a = 71 (RETURN)

Le hemos dicho al ordenador: "Mira, esta a es igual a 71" *y el ordenador guarda en su memoria la pareja*

a = 71

Ahora:

PRINT a (RETURN)

Creo que ya sabias que iba a imprimir el número positivo 71.

¿Qué pasará ahora si hacemos...?

a = 15: PRINT a (RETURN)

Pues que se imprime el número 15.

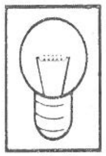
Resulta que a *ha variado de valor. Primero estaba* "ligada" *al valor* 71*, pero después la hemos ligado al valor 15.*

Por eso:

PRINT a (RETURN)

Imprime ahora 15. *Hemos borrado el valor* 71 *anterior y hemos puesto* 15 *con:*

a = 15 (RETURN)



IL6

La memoria es como un armario de cajones. Cada cajón tiene por fuera una etiqueta y dentro puede tener algo o estar vacio.

Cuando hago

a = 71 (RETURN)

estoy guardando en el "cajón" *de etiqueta* a *el número* 71.

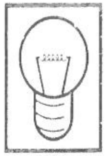
PRINT a (RETURN)

es como abrir el cajón para ver su contenido.

Si ahora hago:

a = 1024 (RETURN)

estoy "quitando" *el valor* "15" *y* "metiendo" *el número* 1024.



IL7

En un cajón o "celda"*, de memoria solo está lo último que entra. Lo que entra,* "echa" *a lo que estaba allí anteriormente.*

Por eso, si ahora:

a = -100 : PRINT a (RETURN)

verás que imprime el número negativo.

-100

Como ves, empleando el signo = *hemos ido variando el valor del número* "ligado"*a la letra* a

¿Como podemos "limpiar" *o* "vaciar" *el cajón* a?

a = Ø (RETURN)

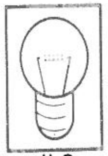
Recuerda que Ø es el número cero y se cruza con una línea para distinguirlo de la letra o.

PRINT a (RETURN)

Como ves, a *contiene cero.*

PRINT n (RETURN)

También n *contiene cero. Claro, no la hemos* "cargado" *con nada de momento. No la hemos usado hasta ahora.*



IL8

> *El* MSX-C *responde con* Ø *si preguntamos por el contenido de un nombre de variable* (a, n,) *cuando no hemos usado antes esa variable o cuando la hemos puesto a cero.*

vamos ahora a cargar a *con* 3,1416 *(Tres enteros y mil cuatrocientas dieciseis diezmilésimas). En el* MSX-C *la coma decimal se expresa con un punto.*

Así:

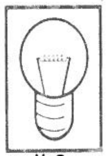
a = *3.1416* (RETURN)

n = 68 (RETURN)

PRINT a,n (RETURN)

Observarás que cada "cajón" *ha conservado lo que le hemos cargado y el valor de* n *no ha modificado el de* a.

Claro, pues son dos variables distintas. Ambas pueden tener y guardar su propio contenido de forma independiente y particular.



IL9

> *Todas las variables se pueden borrar a la vez con* CLEAR.

"Clear" *significa* "limpiar".

Veamos:

PRINT a, n (RETURN)

CLEAR (RETURN)

PRINT a,n (RETURN)

Como ves, después de la instrucción "CLEAR", *el contenido de las variables* a *y* n *es cero. Están* BORRADAS, LIMPIADAS.

ab = 15 : bb = 2001 (RETURN)

PRINT bb, ab (RETURN)

¿Que observamos?

1°.-*Que un nombre de variable puede tener más de una letra o caracter.*

2°.- *Que basta que un caracter sea distinto, para que una variable sea distinta a la otra.*

3°.- *Que cuando se imprimen dos variables en una misma línea, el* MSX-C *las separa, dedicando 14 posiciones de la línea a cada una.*

CLEAR (RETURN)

PESO = 57.300 (RETURN)

PRINT peso (RETURN)

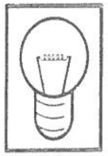
Observa como el MSX-C *suprime los ceros sin valor que sobran.*

Ahora:

PRINT pe (RETURN)

¡¡Que curioso!!

Observa que el MSX-C *imprime lo mismo que en la introducción anterior. Esto quiere decir que 57.3 está en una celda de memoria con la etiqueta* "pe" *y otra con la etiqueta* "peso" *, pero en cambio solo la hemos cargado una vez a* "peso"*, ¿Como es que puede estar en dos variables?*



IL10

> *Los nombres de las variables solo pueden tener uno o dos carácteres. Si tienen más, se ignoran el tercero y los demás como nombre de variable.*

PRINT p, pe, pes, peso (RETURN)

Observa que P *es cero y* PE *es 57.3.*

Pero PES *y* PESO *son como* PE *porque la tercera y cuarta letra se desprecian y queda siempre* PE.

Como ves, P *es cero porqué no la hemos* "estrenado"*. En informática se dice* "Inicializar"*, de iniciar. No le hemos dado ningún valor y el* MSX-C *toma Ø.*

CLEAR (RETURN)

FIN DE LA SESIÓN 3

¿Qué hemos aprendido?

- *Que en la memoria del* MSX-C *se pueden guardar valores numéricos.*
- *Que estos valores numéricos para ser usados mas de una vez, hay que* "cargarlos" *en una variable a través del signo =*
- *Que las variables pueden cambiar su contenido.*
- *Que el* MSX-C *responde con Ø cuando queremos escribir el contenido de una variable que no hemos cargado, inicializado.*
- *Que para indicar decimales se usa el punto, no la coma.*
- *Que todas las variables se pueden borrar a la vez con* CLEAR.
- *Que los nombres de las variables pueden tener como máximo dos caracteres. Si tienen más, se ignoran.*
- *Que el* MSX-C *distribuye las variables escritas con* PRINT *a razón de dos por línea.*

EJERCICIOS SESION 3

Sin consultar el presente manual, resuelve los siguientes ejercicios:

11°.-*Utilizando una sola línea, carga en memoria los valores,* 3, 15, -8400 y 1000 *en las variables* M1, M2, M3 y M4.
En otra línea, ordena al MSX-C *que escriba los valores de* M2 *y* M3.

12°.- *Carga la variable pi con el valor 3,1416*

13°.- *Haz que el* MSX-C *escriba el mensaje:*
"El contenido de la variable M1 es:"

Si no hallas por ti mismo la solución, mírala en la página siguiente y repasa esta sesión.

SOLUCION A LOS EJERCICIOS DE LA SESION 3

11°.- M1 = 3:M2 = 15:M3-8400:M4 = 1000

(RETURN)

PRINT M2, M3 (RETURN)

12°.- Pi = 3.1416 (RETURN)

13°.- PRINT *"El contenido de la variable* M1 *es:"*

(RETURN)

SESION 4

Seguiremos con las variables numéricas.

CLEAR (RETURN)

wm = 8 * 4: PRINT wm (RETURN)

*Observa que se ha escrito el número 32. Has de saber, ya para siempre, que el símbolo * (asterisco), es para el* MSX-C *el signo de multiplicación. Es decir, equivale a la* X *(signo* "por"*) que utilizamos manualmente para multiplicar.*

Por ejemplo:

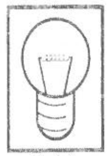
PRINT 5 * 3 * 2 (RETURN)

Fíjate que escribe 30, *que es el resultado de multiplicar 5 x 3 x 2.*

El MSX-C *no puede utilizar el símbolo matemático de multiplicar, la* X, *porque lo confundiría con la letra* X *del abecedario.*

wm = 27/3: PRINT wm "PEPE" (RETURN)

Esto nos resulta familiar.

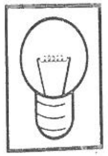

IL11

A una variable numérica se le puede asignar un valor procedente de un cálculo.

Como ves, wm *toma el valor de dividir 27 entre 3.*

En este caso, el MSX-C *realiza primero el cálculo, y carga a la variable el resultado obtenido.*

En este momento es necesario haber comprendido que:


IL12

El simbolo = escrito a continuación de una variable representa que el valor de lo que está a su derecha es asignado (cargado, puesto) a la variable numérica que está a la izquierda de =

Por ejemplo:

C = 2 * 3.1416 * 4: PRINT C (RETURN)

Se imprime el valor 25.1328 que es el resultado de las multiplicaciones de la derecha del símbolo =, símbolo llamado de "asignación", *y ese resultado se asigna a la variable* C.

Vamos a usar cálculos para asignar su valor a diversas variables.

S1 = 5 + 4:S2 = 7 + 28:S3 = 51 + 10 (RETURN)

vamos a imprimir utilizando el punto y coma para separar las variables:

PRINT S1; S2; S3; S4; (RETURN)

Observa que:

1ª.- El punto y coma en PRINT *hace que se escriban los valores uno al lado del otro. Más de dos por línea. En realidad, tantos como quepan en la misma línea.*

2ª.- Los resultados escritos, aunque están juntos, conservan un espacio en blanco para el signo delante y un espacio en blanco detrás. ¿Te acuerdas?.

Ahora utilizaremos la misma PRINT *pero en lugar de separar las variables con un punto y coma, las separaremos solo con una coma. Veamos que pasa:*

PRINT S1, S2, S3, S4 (RETURN)

¿Ves la diferencia?. Ahora solo hay dos números por línea.

La variable S4 *como no la hemos inicializado (cargado), está a cero. Recuerda que en el cálculo a la derecha de =* (llamado expresión

numérica) *está oculto el valor que se va a cargar o asignar a la variable.*

¿Como obtiene el MSX-C *este valor?. Pués realizando los cálculos.*

Repetimos:

PRINT S1; S2; S3; S4 (RETURN)

Veamos que:

S1 *tiene asignado el valor* 9
S2 *tiene asignado el valor* 35
S3 *tiene asignado el valor* 61
S4 *tiene asignado el valor* Ø

¿Qué pasará ahora?

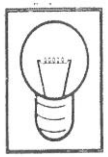
S4 = S1 + S2 + S3 (RETURN)

De momento el MSX-C *sólo dice OK. ¿Que habrá hecho?.*

Vamos a averiguarlo.

PRINT S4 (RETURN)

¡¡Ha sumado S1 + S2 + S3 *y lo ha asignado a* S4!!


IL13

En una EXPRESIÓN NUMÉRICA (conjunto de cálculos a la derecha de =), *pueden estar presentes constantes numéricas* (números) y variables numéricas.

Si ahora hacemos,

PRINT S1; S2; S3; S4 (RETURN)

El valor de S4 ya no es cero, ya que está inicializada con el valor de S1 + S2 + S3

Resumiendo: Cuando el MSX-C *tiene que hacer un cálculo, opera con los números de la expresión. Si dentro de esta expresión se encuentra*

una variable, va a la memoria, toma el contenido o valor de esta variable, y con este valor prosigue el cálculo.

En el ejemplo que hemos visto (S4 = S1 + S2 + S3), *el* MSX-C *ha tomado de la memoria el contenido de* S1, S2 y S3, *los ha sumado, y el resultado lo ha cargado a* S4.

Cambiemos un poco.

ca = S4 * S2 - S4 (RETURN)

PRINT ca

Observa que el MSX-C *ha multiplicado* S4 *por* S2 *y después le ha restado* S4.

En números es: 105 x 35 - 105 = 3570

Ya sabes que también se puede:

PRINT S4 * S2 - S4 (RETURN)

Recuerda que como ultimamente no hemos hecho CLEAR, *tenemos todavia los valores de las variables en la memoria.*

Vamos a escribir un bonito mensaje:

PRINT "la suma de S1 y S3 es:"; S1 + S3 (RETURN)

Observa que:

- *Lo que está entre comillas, se toma como está y se escribe.*
- *S1 + S3, como no está entre comillas, se entiende como un cálculo y el* MSX-C *lo realiza acudiendo a memoria para saber los valores de las variables S1 y S3.*
- *El punto y la coma hace que el mensaje y el resultado del cálculo se escriban próximos.*

CLEAR: CLS (RETURN)

FIN DE LA SESION 4

¿Que hemos aprendido?

- *Que el asterisco* (*) *es el signo de multiplicación en* BASIC.
- *Que una variable puede tener asignado el valor de un cálculo.*
- *Que el símbolo* (=) *no es el símbolo de la igualdad matemática, y que en* BASIC *se utiliza para asignar un contenido o resultado.*
- *Que la derecha del símbolo* (=) *hay una expresión numérica cuyo calor se asigna a la variable numérica cuyo nombre esta a la izquierda del símbolo* (=).
- *Que a una variable numérica puede asignársele un valor procedente de operaciones entre números y variables.*
- *La función o utilidad del símbolo de punto y coma* (;) *en la sentencia* PRINT.

EJERCICIOS DE LA SESIÓN 4

14°.- *Asigna el cálculo* 2 * 5.3 + 15 - 8/2, *cuyo resultado es* 21.6, *a la variable* n.

Haz que el MSX-C *escriba lo siguiente:*

2 x 5,3 + 15 - 8/2 = 21.6
Mensaje *Cálculo*

SOLUCIONES A LOS EJERCICIOS DE LA SESIÓN 4

Este ejercicio puede tener varias soluciones:

1°.- La solución más adecuada,

n = 2 * 5.3 + 15 - 8/2 (RETURN)

PRINT "2 x 5,3 + 15 - 8/2 = "; n (RETURN)

2ª.- Otra, más complicada y menos eficaz,

n = 2 * 5.3 + 15 - 8/2 (RETURN)

PRINT "2 x 5,3 + 15 - 8/2 =" 2 * 5.3 + 15 - 8/2 (RETURN)

Observa que aquí el cálculo se ha hecho dos veces, una al asignarlo a n *y otra al imprimirlo.*

Observa también como en la constante de cadena (entre comillas) *se emplea* x, *pero en el cálculo se cambia por* *.

3ª.- Solución con trampa.

n = 2 * 5.3 + 15 - 8/2 (RETURN)

PRINT "2 x 5,3 + 15 - 8/2 = 21.6" (RETURN)
Copiado del ejercicio.

SESIÓN 5

Vamos a ver ahora algunas operaciones aritméticas.

SUMA

su = 3 + 7 (RETURN)

PRINT su (RETURN)

La variable SU *ha tomado el valor* 10, *que es el resultado de la operación de suma.*

Así, SU *contiene ahora el valor* 10.

RESTA

re = 4096 - 512 (RETURN)

PRINT re (RETURN)

Ahora, la variable RE *ha tomado el valor* 3584, *que es el resultado de la operación de resta.*

SUMAS Y RESTAS

V1 = RE + SU - 1000 (RETURN)

PRINT v1 (RETURN)

Observa como en la expresión numérica anterior hay variables numéricas y constantes numéricas.

Las variables numéricas son RE *y* SU. *La constante es* -1000.

Para poder realizar el cálculo el MSX-C *acude a la memoria para tomar el contenido de cada una de las variables. En este caso suma el valor de* RE *al de* SU *y luego les resta* -1000.

MULTIPLICACIÓN

mu = re * su (RETURN)

PRINT mu (RETURN)

Exactamente igual que en el caso anterior, pero ahora en multiplicación. Recordemos que el asterisco (*) es, en Basic, el signo de multiplicación.

DIVISIÓN

PRINT re/su (RETURN)

Es también el momento de recordar que el símbolo (/) es, como en matemáticas, el de la división.

También es oportuno recordar que se puede imprimir el resultado de un cálculo sin tener que guardarlo en una variable.

PRINT "re/su ="; re/su (RETURN)

Ya sabes que la diferencia entre una constante de cadena y cualquier otra cosa es que la constante de cadena va entre comillas, y se escribirá tal como está.

Por eso "RE/SU" *no se cálcula y* RE/SU *sí.*

SUMAS, RESTAS, MULTIPLICACIONES Y DIVISIONES

PRINT 53 + 15 - SU + 35 * RE - 2525/12.5
(RETURN)

No estaría mal que tú mismo hicieras también el cálculo a mano.

¿Como lo hace el MSX-C?

- Primero calcula las multiplicaciones y las divisiones.

35 * RE = 35 * 3584 = 125440
2525/12,5 = 202

- Después las sumas y las restas.

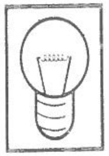
53 + 15 -10 + 125440 - 202 = 125296

Recordemos: Primero se ejecutan siempre las multiplicaciones y las divisiones. Después se ejecutan las sumas y las restas.

SU = SU + 1000 (RETURN)

PRINT SU (RETURN)

Aquí podemos comprobar y recordar algo importante. El signo (=) de asignación no es el signo de "IGUAL" *que se emplea en matemáticas.*



> *El signo (=) de asignación adjudica el valor de la expresión numérica de la derecha, a la variable de la izquierda.*

IL14

Repasemos otros conceptos.

CLEAR (RETURN)

Ya sabes que acabas de borrar (o limpiar, o dejar a cero), todas las variables.

a = 10 (RETURN) *(inicializamos* a*)*

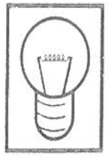
a = a + 10 (RETURN) *(sumamos* 10 *a* a*)*

PRINT a (RETURN) *(vemos el valor de* a*)*

Si ahora subimos el cursor hasta la línea que suma 10 *a* a (a = a + 10) *y hacemos* (RETURN) *dos veces, veremos que hemos vuelto a sumar* 10 *al valor anterior de* a. *Para comprobarlo, haz*

PRINT a (RETURN)

Lo que hemos hecho ha sido un BUCLE *o* CICLO *de sumas al valor de* a.



El valor de una variable puede figurar como parte de la asignación a esa misma variable.

IL15

Este era el caso de a = a + 10, *donde la variable* a *está a la izquierda y a la derecha del signo* (=) *de asignación.*

Para entenderlo, podemos hablar de valor VIEJO *de la variable y de valor* NUEVO *de la variable.*

Por ejemplo: B5 = B5 * 13 *significa que la* "nueva" *B5 vale lo que valía la* "vieja" *B5 antes, multiplicado por 13.*

a = a + 10 *quiere decir que la* "nueva" *a toma el valor de la* "vieja" a, *más* 10.

Veamos una nueva operación aritmética.

EXPONENCIACION

Cuando un número se multiplica varias veces a si mismo, se dice que lo estamos "ELEVANDO" *a una potencia determinada.*

CLEAR (RETURN) (borrar variables)

(SHIFT) + (HOME) (borrar pantalla)

a = 4 (RETURN) (inicializamos a con el valor 4)

a = a * 4 (RETURN) (multiplicamos a por su primer valor)

PRINT a (RETURN) (vemos su valor actual)

(⇧) (subir hasta a = a * 4)

(RETURN)

(RETURN)

PRINT a (RETURN)

Comprobamos que ahora el valor de a *es de* 64.

El MSX-C *puede hacer automaticamente este cálculo de la potenciación de una variable, mediante el símbolo del acento circunflejo (^).*

Veamos:

CLEAR (RETURN)

a = 4:PRINT a^3 (RETURN)

¿Ves?. El resultado es el mismo de antes. El MSX-C *ha* "ELEVADO" *a* a *a la tercera potencia.*

PRINT 1000^3 (RETURN) (esto da mil millones).

PRINT 3.1416 * 5^2 (RETURN) (esto calcula la superficie de una circunferencia de radio 5).

CLEAR: cls (RETURN)

En una expresión numérica, la exponenciación (^) es la primera operación que realiza el MSX-C.

Ejemplo:

P = 35 + 15/3-2^3 + 5 * 8 (RETURN)

PRINT p (RETURN)

Esta expresión numérica, el MSX-C *la calcula así:*

1°.- Las exponenciaciones

2^3 = 2x2x2 = 8

2°.- Las multiplicaciones y las divisiones

15/3 = 5

5 * 8 = 40

3°.- Las sumas y las restas

35 + 5-8 + 40 = 72

Siempre por este orden:

POTENCIAS - MULTIPLICACIONES y DIVISIONES - SUMAS y RESTAS.

FIN DE LA SESION 5

¿Que hemos aprendido?

- *Sumas, restas, multiplicaciones, divisiones y exponenciaciones en el* MSX-C.
- *Variables cuyo valor dependen de otras variables.*
- *Variables influidas por su antiguo valor.*
- *Que el símbolo (=) no es la igualdad matemática. Es el símbolo de asignación de valor.*
- *Hacer un bucle repitiendo las sentencias con ayuda del cursor hacia arriba.*
- *El orden de cálculo de las expresiones numéricas.*

EJERCICIOS DE LA SESION 5

15°.- *Ejecuta, a través del uso de* PRINT, *el siguiente cálculo:*

7 + 15 x 2 - 14/2

16°.- *Haz que la variable* c *sea al principio igual a 8, y después que se multiplique a si misma.*

Muestra el resultado con PRINT c.

17°.- *El número pi es 3.1416.*

La fórmula de la longitud de una circunferencia es 2 x pi x radio.

Calcula e imprime la longitud de una circunferencia de radio 225.

SOLUCION A LOS EJERCICIOS DE LA SESION 5

15°.- PRINT 7 + 15 * 2 - 14/2 (RETURN) (= 30)

16°.- c = 8 (RETURN)

c = c * c (RETURN)

PRINT c (RETURN) (= 64)

17°.- PRINT 2 * 3.1416 * 225 (RETURN) (= 1413,522)

SESION 6

Conocemos ya las variables numéricas. Algo así como etiquetas de cajones en las cuales guardamos provisionalmente algunos valores numéricos.

En algunos casos guardamos los valores directamente:

a = 35 : PRINT a (RETURN)

En otros casos guardábamos los valores a través de cálculos.

b = a ˆ 3 : PRINT a, b (RETURN)

Ahora es el momento de que aprendas a preguntar al MSX-C *por el valor o contenido de una variable sin tener que teclear* PRINT.

Primero tienes que saber que para teclear ?, *debe hacerse* (SHIFT) + (?)

Ahora, haz esto:

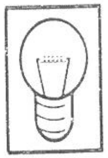
? a (RETURN)

Escribir ? *es igual que escribir* PRINT

Cuando escribimos ? *es como si le dijéramos al* MSX-C. . . . *¿Qué es . . .?*

? b (RETURN)

¿ves que fácil?.



IL16

? *es una forma breve de escribir* PRINT.

? a * b (RETURN)

Bien, ya lo has entendido, ¿verdad?.

Vamos a ver ahora otro tipo de variables, las variables de cadena de caracteres. Haz:

di = "lunes" (RETURN)

Fijate que el MSX-C se queja. Nos dice:

TYPE MISMATCH (error de teclado).

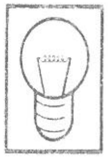
Por alguna razón, di *no puede ser variable de cadena. (no puede guardar letras).*

Ahora, haz:

di$ = "lunes" (RETURN)

Verás como ahora el MSX-C *dice ¡OK!*

? di$ (RETURN)



IL17

Los nombres de variables de cadena tienen uno o dos caracteres, pero han de ir seguidos por el símbolo del dólar ($).

di$ *es una variable de cadena de caracteres.*

di *es una variable numérica.*

di = 1 (RETURN)

?"el dia número"; di; *"de la semana es ";* di$ (RETURN)

Si lo analizas, fijate que el ? (PRINT) escribe primero

El dia número

A continuación, con di *escribe:*

1

después escribe la constante de cadena

de la semana es

y con di$ *escribe*

lunes

En esta ? (PRINT) *hay dos textos mezclados con dos variables. Una de ellas,* di, *es numérica y la otra,* di$, *es de cadena.*

di = 4 : di$ = "jueves" (RETURN)

Ahora, con el cursor hacia arriba, vete a la línea que tiene la sentencia de impresión ? "El dia . . . . . . y

(RETURN)

Como hemos alterado el valor de di *y de* di$ *y hemos impreso una nueva frase.*

Así, cuantas veces quieras, puedes ir cambiando el valor de di *y* di$ *y obtendrás los mensajes apropiados.*

Claro que si le dices algo erróneo, por ejemplo:

di = 3 : di $ = "sábado" (RETURN)

y ahora vuelves a ir a la ? para pedir que te escriba el mensaje, y el MSX-C *te escribirá:*

"El dia número 3 de la semana es el sábado"

lo cual es erróneo, pero el MSX-C *hace lo que tú le has pedido.*

¿Te das cuenta que el MSX-C *solo hace aquello que tú le dices?. El no sabe que el tercer día de la semana es el miércoles. Tan solo* "TIENE" *lo que tu le "das".*

También puedes hacer que el MSX-C *escriba cosas absurdas.*

di $ = "bélgica, la hermosa" (RETURN)

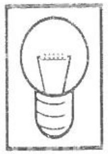
vuelve a situarte con el cursor en la sentencia de impresión (?. . .) *y*

(RETURN)

El MSX-C *escribirá:*

El dia número 3 de la semana es bélgica, la hermosa

Como ves, en di $ *puedes* "cargar" *todo lo que te apetezca, siempre que vaya entre comillas.*



IL18

Tener lógica quiere decir que cuando se hace un programa hay que pensar en lo que deseamos que haga, para que no hagas cosas absurdas.

FIN DE LA SESION 6

¿Que hemos aprendido?

- *La ? es una abreviatura de* PRINT
- *Las variables de caracteres de cadena se nombran con un signo de dólar* ($) *al final.*
- *En las* PRINT *puede haber mensajes, variables de cadena, variables numéricas y cálculos.*
- *Las variables de cadena se cargan con caracteres situados entre comillas.*
- *Cuando* "CARGAMOS" *variables hemos de considerar que lo sean con valores adecuados para lo que queremos obtener.*

EJERCICIOS DE LA SESION 6

18°.- *Inicializa la variable* ed *con tú edad. Inicializa la variable* no$ *con tu nombre. Imprimir:*

"Soy (nombre) y tengo (edad) años"

19°.-  di = 5 : xy$ = "viernes" (RETURN)

Imprimir:

HOY ES VIERNES, DIA 5

SOLUCIONES A LOS EJERCICIOS DE LA SESION 6

18°.- *Por ejemplo:*

ed = 15 : no$ = "JOAN CREUS" (RETURN)

? "Soy "; no$; " y tengo "; ed; " años" (RETURN)

19°.- ? "Hoy es "; xy$;", dia "; di (RETURN)

SESION 7

Hoy vamos a dar un salto de gigante.

Ya sabes que en la memoria del MSX-C *puedes guardar datos numéricos y de cadena. Tan solo tienes que darles nombre para poderlos cambiar, representar o usar.*

Pero, ¿Que haces si...?

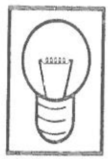
a = 5 (RETURN)

Introduces un dato

y ¿cuando haces...?

? a (RETURN)

El MSX-C *obedece una sentencia*



IL19

Las órdenes, sentencias que tecleamos, también pueden introducirse en la memoria.

¿Como?. Vamos a verlo

? a (RETURN)

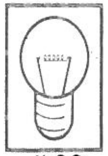
Esto, ya lo sabes, hace que el MSX-C *escriba en pantalla el valor de la variable* a.

Pero ahora...

10 ? A (RETURN)

¡Caramba! ¿que ha pasado?

El MSX-C *parece no hacer nada. Ni tan siquiera ha dicho* OK.



IL20

> *Cuando una sentencia (introducción, orden) va precedida por un número entero, no se ejecuta. Se guarda en memoria con ese número de identificación.*

Para comprobar que realmente 10?A *se ha almacenado en la memoria, vamos a listarla, a llamarla y traerla desde la memoria a la pantalla.*

LIST 10 (RETURN)

Fíjate, aquí está. Fíjate más, el MSX-C *ha cambiado el* ? *por su equivalente* PRINT

5 a = 71 (RETURN)

Ya hemos almacenado esta sentencia también en la memoria.

Hemos introducido el número 71 *en* a. *El número* 5 *no es un valor, es un identificativo, un número de orden.*

Veamos como se ha introducido en la memoria.

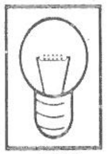
LIST 5 (RETURN)

Aquí está también.

Atención ahora, vamos a descubrir algo muy importante:

LIST (RETURN)

Se han listado las dos sentencias que habíamos almacenado en la memoria precedidas con un número y en orden ascendente del número que llevan.



IL21

> *Si se* LIST (RETURN) *, todas las sentencias que hemos introducido, se listan por el orden de su número.*

¡¡Ya tenemos un programa en memoria!!

Vamos a poner otra sentencia en la memoria.

2 CLS (RETURN)

Ya la tenemos en memoria.

limpiemos la pantalla

(SHIFT) + (HOME)

Y ahora volvamos a listar todas las sentencias que hemos introducido.

LIST (RETURN)

Ya tenemos a la vista nuestro primer programa. *Veamos que dice:*

2	CLS	*(limpiar la pantalla)*
5	A = 71	*(cargar* 71 *en* a*)*
10	PRINT a	*(Imprimir* a*)*

Ahora queremos que este programa (conjunto de tres sentencias) se ejecute. Para ello, tenemos que dar la orden RUN .

RUN (RETURN)

¡¡Nuestro programa ha funcionado!!

Veamos que ha hecho.

1° *Se ha borrado la pantalla*
2° *Se ha cargado* 71 *en* a
3° *Se ha escrito el valor de* a (71)

Exactamente lo que teníamos ordenado en las sentencias.

A este conjunto de sentencias que se ejecutan seguidamente, una detrás de otra en el orden de sus números de identificación, le llamamos programa

Por cierto, ¿estarán todavía en memoria las instrucciones?

Vamos a comprobarlo

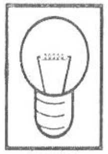
LIST (RETURN)

Efectivamente, sigue estando en memoria.

Veamos si vuelve a funcionar.

RUN RETURN

Sí, todavia funciona.


IL22

> RUN *se utiliza para ordenar al* MSX-C *que lleve a cabo las instrucciones del programa que tenga en memoria.*

Fíjate que el número de instrucción (número de identificación), es lo que sirve para que el MSX-C *sepa en que orden tienen que ejecutarse.*

LIST RETURN

Observa que el MSX-C *ha pasado todos los carácteres a mayúsculas. Esto no tiene ninguna importancia, solo sirve para hacer más claros los listados de programas.*

Si queremos borrar las variables en memoria, como ya lo hemos hecho anteriormente muchas veces, haz

CLEAR RETURN

¿Se habrá borrado también nuestro programa?

Veámoslo

LIST RETURN

Pues no, no se ha borrado nuestro primer programa. No se ha borrado porqué CLEAR *no borra los programas, solo borra las variables.*

Para borrar programas se utiliza la sentencia NEW

Probemos

NEW RETURN

Veamos si sigue el programa allí.

LIST RETURN

Ahora el programa ya se ha borrado, ya no está en memoria. Así pues, recuerda que NEW *sirve para borrar los programas que estén en la memoria.*

NEW *significa nuevo, otro, y borra totalmente el programa que haya en memoria para dejar sitio a otro programa.*

10 di = 7 RETURN

20 di$ = " domingo " RETURN

30 ? "El dia "; di; " es el "; di$ RETURN

Observa que en vez de ejecutar cada vez la sentencia que escribimos, el MSX-C *esperará a ejecutarlos todos y por orden, a que le demos la orden* RUN.

¡¡Adelante!!

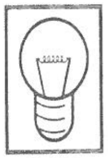
RUN RETURN

Ahora prueba

F5

también ejecuta el programa.

Pruebas varias veces el F5


IL23

> *La tecla* F5 *tiene el poder de poner en marcha* (RUN) *el programa que haya en memoria y nos ahorra el tener que escribir* RUN.

F5 *equivale a*

RUN (RETURN)

Es pues más cómodo utilizar F5 *para* "lanzar" *o ejecutar programas.*

FIN DE LA SESION 7

¿Que hemos aprendido?

- *En memoria puede estar:*
  variables numéricas
  variables de cadena
  instrucciones de programa
- *Las instrucciones, para estar en memoria, tienen que estar numeradas.*
- *El número de instrucción (identificativo) indica al* MSX-C *el orden en que deben ejecutarse.*
- LIST *(número) lista en la pantalla la instrucción que tenga ese número.*
- LIST, *lista en pantalla todas las instrucciones de un programa.*
- NEW *borra programas.*
- RUN *ejecuta (pone en marcha) el programa que esté en memoria.*
- (F5) *hace las funciones de* RUN.

EJERCICIOS DE LA SESION 7

20°.-*Introducir en memoria el siguiente programa:*

NEW (RETURN)

10 a = 16 : b = 4000 (RETURN)

20 su = a + b (RETURN)

30 di = b/a (RETURN)

Introducir una nueva instrucción con el número 40, *que ordene imprimir* su *y* di

Listar el programa

Ejecutar el programa con tecla de función F5

21°.-*Añadir una sentencia, al principio del programa, que borre la pantalla antes de escribir el resultado. Ponerle el número* 5 *a esa instrucción.*

Después ejecutar con

RUN (RETURN)

22°.-*Con el programa tal como está, probar varias veces*

(F5)

y explicar lo que sucede.

20º.- 40 PRINT suma, división (RETURN)
(o 40 PRINT su, di)

LIST (RETURN)

(F5)

21º.- 5 CLS (RETURN)

RUN (RETURN)

22º.-*Como* CLS *borra cada vez la pantalla, cada vez el resultado se escribe en el mismo sitio.*

SESION 8

Hasta ahora hemos aprendido, entre otras cosas, a imprimir constantes de cadena, constantes numéricas e incluso variables de estos dos tipos.

Repasemos:

10 JU = 1.80 (RETURN)

20 no$ = "JUAN" (RETURN)

30 PRINT "SOY "; no$; " y mido "; JU; " metros"
(RETURN)

Si ahora hacemos

(F5)

veremos que el MSX-C *imprime (escribe) en la pantalla, por este orden:*

1º.- *La constante de cadena* "SOY"

2º.- *El contenido de la variable de cadena* no$ *que, en este momento, es* "JUAN", *tal como la hemos cargado en la sentencia* 20.

3º.- *La constante de cadena* "y mido".

4º.- *El contenido de la variable numérica* JU *que hemos cargado en la instrucción nº* 10 *con el valor* 1,80.

5º.- *La constante de cadena* "metros".

Los puntos y comas hacen que constantes y variables se escriban juntas. Hay que cuidar los espacios en blanco entre constantes y variables de cadena. Ya sabes que las constantes y variables numéricas se escriben con un espacio o, signo por delante, y un espacio por detrás.

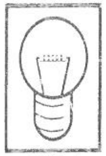
(SHIFT) + (HOME)

Vamos a aprender a modificar un programa. Para ello, para escribir de nuevo en la pantalla el programa que tenemos en memoria, utilizaríamos la sentencia LIST, *pero lo haremos ahora apretando la tecla* F4, *que nos va a realizar la misma función.*

(F4) (RETURN)

Rapidamente aparece el listado del programa. Para algo que entenderás más adelante, vamor a borrar el OK. *Para ello:*

(⇧) (DEL) (DEL)



IL24

> *La tecla* DEL *borra, carácter a carácter, cada vez que se teclea.*

ATENCION:

(F5)

Aparece la palabra RUN *se ejecuta el programa y termina con un* OK

Para modificar (cambiar) algo del programa, subamos con (⇧) *hasta la línea número* 10 *del programa. Para ello:*

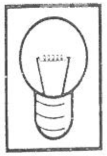
(⇧) *(7 veces)*

(⇨) *(8 veces)*

Ahora el cursor debe estar sobre el 8 *del valor de la variable JU. Si no lo está, mueve el cursor hasta que se situe* exactamente *sobre el número* 8 *de:* 10 JU = 1.[8]

6

¡El número 8 *ha cambiado por el* 6!



IL25

> *Para cambiar el valor de cualquier carácter de la pantalla, basta situar el cursor encima de la posición a cambiar, y teclear el nuevo carácter deseado.*

Falta todavia algo:

(RETURN)

Al presionar RETURN *estamos modificando el programa que habia en memoria:*

Antes era: 10 JU = 1.8

Ahora debe ser: 10 JU = 1.6

Vamos a ver si es verdad. Baja el cursor hasta la línea que dice RUN

(⇩) *(3 veces)*

(RETURN)

El programa se ha ejecutado pues el RETURN *que hemos dado estaba sobre la palabra* RUN

Comprueba que JUAN mide ahora 1,60 *metros, de acuerdo al cambio que hemos hecho en el programa.*

Otro caso:

(⇧) *(7 veces)*

(⇨) *(8 veces)*

Debes estar situado sobre el número [6] *de la instrucción* 10.

75 (RETURN)

(⇩) *(3 veces)*

Ahora hemos vuelto a modificar el programa y el cursor debe estar sobre la palabra RUN

(RETURN)

Observa como ha cambiado la impresión.

Incluso se ha alargado la frase, pues hemos incluido 1,75 *donde antes habia* 1,6.

Cuando una frase es más larga que la que habia anteriormente en su lugar, no tenemos problemas, pero en el caso contrario, si la nueva frase que escribimos es más corta que la anterior, podemos tener algún problemilla.

Probemos a ver que pasa si volvemos a hacer JU = 1.6

(⇧) *(7 veces)*

(⇨) *(8 veces)*

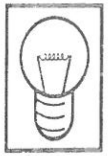
(DEL)

El "7" ha desaparecido y el cursor está sobre el "5".

6 (RETURN)

(⇩) *(3 veces) y* (RETURN)

Como ves, la última letra de la frase anterior, la letra s *se repite, pues al ser más corta la nueva frase, la palabra* "metros" *retrocede pero no borra lo que había antes.*



IL26

Para evitar que queden restos de impresiones anteriores allí donde vamos a imprimir algo nuevo, es conveniente borrar lo anterior.

Para ello hay que saber y controlar donde se va a imprimir.

El MSX-C *tiene la pantalla dividida en* 24 *líneas (numeradas del* Ø *al* 23*) y* 37 *columnas (numeradas del* Ø *al* 36*).*

Hay una instrucción (LOCATE) *que indica el punto exacto donde debe situarse el cursor. Por ejemplo, para iniciar una impresión.*

25 LOCATE 5,20 (RETURN)

(F5)

Como ves, aparece la impresión en la fila 20, *comenzando en la columna* 5.

O sea, que:

LOCATE columna, fila

indica al cursor que inicie la impresión en la columna y fila indicadas.

Ahora, con las teclas que mueven el cursor, vete a la línea 10 *del programa en la pantalla. Para ello has de hacer, más o menos:*

(⇧) *(21 veces)*

(⇨) *(8 veces)*

95 (RETURN)

Hemos dado a JU *el valor* 1.95

Ahora ejecuta sobre RUN

☞ (⇩) *(3 veces) y* (RETURN)

El programa se ha ejecutado y la impresión se ha ajustado a la nueva longitud.

Fíjate que como estamos variando la variable JU, *en consecuencia varía también la impresión de salida.*

Observa que LOCATE *hace aparecer en el sitio indicado la impresión.*

Vamos ahora a entrar dos nuevas líneas de programa, con un número de orden o secuencia situado entre dos instrucciones ya existentes:

☞ 26 PRINT STRING$ (31, " ") (RETURN)

☞ 27 LOCATE 5,20 (RETURN)

Ya hemos introducido dos nuevas líneas de programa.

STRING$ (31, " ") *es una instrucción que* "entrega" *al PRINT una cadena de espacios* (31 en este caso) *en blanco. Si pusieramos* STRING$ (12, "F") *entregaría una cadena de* 12 *efes.*

Con esta instrucción, al imprimir una cadena de 31 *espacios en una línea, es como si borráramos lo que había en la línea.*

Veamos el programa en su totalidad.

☞ (SHIFT) + (HOME) (F4) (RETURN)

En pantalla sale:

10 JU = 1.95 *Aquí se carga* JU *con 1.95*

20 NO$ = "JUAN" *Se carga* NO$ *con* "JUAN"

25 LOCATE 5,20 *El cursor, aunque no se vea, se posiciona en la línea* 20 *y columna* 5.

26 PRINT STRING$ (31, " "): LOCATE 5,20

Se imprime 31 *espacios en blanco. Lo que hubiera antes aquí, se ha borrado. Luego, el cursor vuelve donde estaba.*

30 PRINT "SOY "; NO$; " y mido"; JU; " metros"

Aquí se ordena escribir el mensaje deseado.

Como ves, hemos ido introduciendo líneas o instrucciones nuevas de programa entre otras ya existentes. La única condición, es que tengan un número de instrucción propio y único, que corresponda al lugar donde queremos que vaya.

Por ejemplo, a la instrucción número 26 le hemos dado este número porqué queríamos que se ejecutara después de la 25 *y antes de la* 30.

Puede darse el caso de que entre dos líneas existentes no podamos situar otra, por ser las existentes consecutivas.

Por ejemplo, entre las líneas 25 *y* 26 *no podemos colocar ninguna instrucción.*

¿Que podemos hacer si entre la instrucción 25 *y la* 26 *queremos colocar una instrucción?*

El MSX-C *tiene una solución:*

☞ RENUM (RETURN)

☞ LIST (RETURN)

Observarás que RENUM *deja el programa como estaba pero renumera las líneas de* 10 *en* 10 *para dejar sitio por si queremos poner otra sentencia. Por ejemplo:*

antes: 25 LOCATE }
26 PRINT } No cabia otra instrucción entre las dos.

ahora: 30 LOCATE }
40 PRINT } cabes varias instrucciones entre las dos.

☞ 35 CLS (F5)

Con esta instrucción el programa se ejecuta igual que antes pero se borra la pantalla antes de imprimir.

(F5) *(varias veces)*

Observa que cada vez se ejecuta de la misma manera.

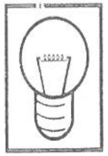
(F4) (RETURN)

Hay una línea de instrucción con el número 35

RENUM (RETURN)

(F4) (RETURN)

Fíjate que ya no existe la línea 35 *pero que el programa es el mismo.*



IL27

> *Lo importante no es el número de sentencia, sinó la secuencia u orden de las sentencias.*

Con RENUM *podemos hacer que el programa comience en un número de sentencia cualquiera (menor que 65530).*

Por ejemplo:

RENUM 1000,,7 (RETURN)

Esto hace que el programa se renumere y empieze a partir del número 1000, *y después las numera de 7 en 7.*

Para verlo:

(F4) (RETURN)

Para comprobar que el programa funciona igual.....

(F5)

¿De acuerdo?

FIN DE LA SESION 8

¿Que hemos aprendido hoy?

- DEL *borra el caracter donde está el cursor.*
- *Cambiar un caracter.*
- *Cambiar el valor de una variable con el cursor y repetirlo varias veces.*
- *Que conviene borrar primero allí donde se va a escribir otra vez.*
- *El uso de* LOCATE.
- *Introducir líneas nuevas en un programa.*
- *Un uso de* STRING$ (.....)
- *El uso de* RENUM, *comienzo, salto.*
- *Que lo importante es el orden de las sentencias.*

EJERCICIOS DE LA SESION 8

23°.- *Renumerar, desde el número* 1, *el programa que está en memoria al acabar la sesión.*
Listarlo para certificar que la nueva numeración y ejecutarlo para comprobar que funciona.

24°.- *A continuación hacer que en vez de imprimirse la salida en la línea* 20, *lo haga en la línea* 12.

Una ayuda: - mover el cursor
- cambiar 20 *por* 12 *y* RETURN

Ejecutar con F5. *(Cuidado, el* RUN *debe darse sobre una línea* "limpia"*)*

25°.- *Renumerar el programa desde* 10 *y de* 10 *en* 10.
Comprobarlo.

26°.- *Cambiar el nombre de* JUAN *por* ELISA *y la estatura a* 1.72
Ejecutar.

SOLUCIONES A LOS EJERCICIOS DE LA SESION 8

23°.- RENUM 1,,1 (RETURN)

(F4) (RETURN)

(F5)

24°.- (↑) *(4 veces)*

(→) *(11 veces)*

12 (RETURN)

(HOME) (F5)

25°.- RENUM (RETURN) (F4) (RETURN)

26°.- (↑) *(9 veces)* (→) *(8 veces)*

72 (RETURN)

[⇨] *(8 veces) y* Elisa" [RETURN]

[HOME] [F5]

SESION 9

Ya hemos aprendido a modificar un programa en memoria. También, a estas alturas, ya sabemos imprimir en la pantalla un mensaje, con constantes y variables, en cualquier punto de la pantalla.

Hoy vamos a aprender a introducir datos en un programa desde el teclado.

[F2] [RETURN]

Verás que aparece la palabra AUTO *y a continuación el número* 10. *El cursor se queda a continuación del* 10, *esperando.*

¿Qué espera?. Espera a que le entremos una instrucción en BASIC MSX.

KEY OFF: CLS [RETURN]

Aparece el número 20 *para otra instrucción.*

LOCATE 5,12 [RETURN]

Ahora la 30....

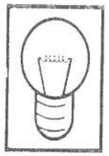
PRINT "Hola Pepe"

Ya tenemos un programa, pero el MSX-C *nos pide otra línea.*

¿Como haremos para salir de esta numeración automática de líneas?

[CTRL] + [STOP]

Ya está.



IL28

[F2] *Tiene la función auto que sirve para ir numerando automaticamente las sentencias de programa.* AUTO *se termina con* [CTRL] + [STOP]

Ahora:

(F5)

Como ves, el programa funciona perfectamente.

(F4) (RETURN)

Vamos a modificar el programa sustituyendo Pepe *por una variable de cadena.*

(⇧) *(2 veces) y* (⇨) *(14 veces)*

Ahora el cursor está entre HOLA *y* PEPE.

(DEL) *(5 veces)*

ha desaparecido el nombre.

(INS)

(BARRA ESPACIADORA) *(Barra blanca de espacios)*

Como ves hemos introducido, insertado, un espacio en blanco al final de la palabra HOLA.

(⇨) *(una vez)* ; NO$ (RETURN)

Aquí lo que hacemos es decirle al MSX-C *que imprima la palabra* HOLA *y el contenido de la variable de cadena* NO$. *El espacio que hemos puesto a continuación de* HOLA *es para separar la palabra* HOLA *de las que puedan venir a continuación, pues ya sabes que el punto y coma (;) hace que las impresiones de cadena salgan juntas.*

(SHIFT) + (HOME) (F5)

Observarás que solo imprime "Hola". *Ello es así porqué si miras bien el programa verás que en ningún momento hemos cargado la variable* NO$. *En consecuencia el* MSX-C *no imprime nada a través de* NO$.

LIST (RETURN)

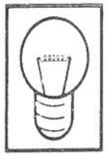
25 INPUT "cual es tu nombre " no$ (RETURN)

(F5)

¡¡El MSX-C *te está preguntando cual es tu nombre!!*

(escribe tu nombre) (RETURN)

¡¡el MSX-C *te saluda!!*



IL29

La sentencia INPUT *emite un mensaje y queda a la espera de que entres algún dato por teclado.*

Fíjate que la línea 25 *de* INPUT *emite el mensaje que hemos puesto y carga la variable* no$. *Luego, el programa se sigue ejecutando.*

LOCATE *ha situado el mensaje del* INPUT, *el cual, por cierto, como el* MSX-C *nos pide a través de una* INPUT *que le entremos siempre algún dato, siempre termina con una interrogación. (?). Después de esta* INPUT, *la instrucción* PRINT *se ejecuta en la línea siguiente.*

Vamos a mejorar el funcionamiento del programa.

(F4) (RETURN)

Modifiquemos la sentencia 30 *para que el* PRINT *salga donde salía antes.*

(⇧) *(2 veces) y* (⇨) *(2 veces)*

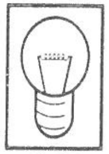
(INS) LOCATE 5,12: (RETURN)

Con (INS) *podemos introducir lo que queramos, en el lugar donde esté el cursor. (A esto se le llama,* INSERTAR*)*

La señal que nos indica que podemos insertar o introducir caracteres, es el cursor que se queda pequeño.

Si nos equivocamos y tenemos que borrar o retroceder, se pierde la posibilidad de insertar. En este caso, habría que apretar otra vez

[INS]



IL30

> *La facilidad de insertar es posible solo cuando el cursor es como una raya. Para insertar tiene que apretarse la tecla* [INS] *y después teclear lo que queremos insertar.*

Bien. Hemos hecho

30 LOCATE 5,12:PRINT "HOLA";NO$

Ahora:

[F5]

JUAN [RETURN]

Pues no. No está bien.

No hemos borrado la línea antes de imprimir. Para borrar, lo hemos aprendido en la sesión anterior. ¿Te acuerdas?.

Para hacerlo...

[F4] [RETURN]

26 LOCATE 5,12:PRINT STRING$(31, " ")

[F5]

(tu nombre) [RETURN]

Muy bien. La sentencia 26 hace que se borre la línea de la pantalla para la impresión siguiente y todo queda ahora bien.

Para que veas que un ordenador hace solo lo que tiene programado, haz:

[F5]

no se [RETURN]

Hagamos un par de mejoras en el programa. Observa que la línea 20 *hace que el* INPUT *se coloque. Vamos a poner el* LOCATE *en la misma línea del* INPUT.

[F4] [RETURN]

[⇧] *(4 veces)* [⇨] *(2 veces)*

Ahora el cursor está entre el n° 25 *e* INPUT

[INS] LOCATE 5,12: [RETURN]

Ahora la línea 20 sobra pues dos LOCATE 5,12 *seguidos no tienen sentido. La colocación del* INPUT *ahora se controla en la sentencia* 25.

Para borrar la sentencia n° 20 *del programa (nos sobra), nos vamos a una zona libre de pantalla, es decir, ponemos el cursor en una línea que no haya nada escrito, y escribimos el número de la sentencia que queremos eliminar, en este caso el* 20.

[HOME] 20 [RETURN]

El MSX-C *al detectar una línea de programa sin contenido, la borra si ya existia.*

Comprobémoslo:

LIST (RETURN)

Ahora tan solo nos queda poner una numeración más bonita a nuestro programa.

Para ello:

RENUM (RETURN)

(SHIFT) + (HOME) (F4) (RETURN)

FIN DE LA SESION 9

¿Que hemos aprendido?

- AUTO, *numera automaticamente las líneas de un nuevo programa que vamos a entrar.*
- AUTO *comienza con* (F2) *y termina con* (CTRL)+(STOP)
- *Insertar caracteres para modificar una sentencia.*
- *Insertar se hace con* (INS) *y el cursor se vuelve pequeño.*
- INPUT *admite datos desde el teclado y puede emitir una pregunta orientativa.*
- *Sabemos borrar una línea de programa.*

EJERCICIOS DE LA SESION 9

27°.- *Realiza un programa en el que cada instrucción haga lo siguiente:*

10.- *Borra la línea de teclas de función y la pantalla.*

20.- *Que nos pida por pantalla un número.*

La variable de este número la llamaremos nu *en el programa.*

La pregunta ha de salir en la columna 5 *de la línea* 5 *de la pantalla.*

30.- *Borrar la pantalla.*

40.- *Imprimir en la columna Ø de la fila 12, el resultado de multiplicar por si mismo el número que hemos entrado, con un mensaje así:*

"El cuadrado de (aquí el número) es:" (aquí el resultado).

```
    NEW
10  KEY OFF: CLS
20  LOCATE 5,5: INPUT "NUMERO"; NU
30  CLS
40  LOCATE 0,12:PRINT "El cuadrado de "; NU; " es "; NU*NU
```

Haciendo varias veces (F5) *puedes practicar.*

SESION 10

Vamos avanzando. Pero ten en cuenta que sobre lo que hemos visto volveremos con frecuencia pues las sentencias que ya hemos expuesto pueden hacer aún más cosas de las que hemos visto.

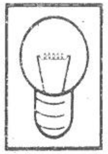
De vez en cuando volveremos sobre lo anterior para ver algún aspecto nuevo.

En esta sesión veremos una sentencia muy importante: GO TO. *En Inglés significa:* IR A....

Como sabes, el programa se ejecuta ordenadamente siguiendo el número de línea del programa.

Hay algunas sentencias, entre ellas GOTO, *que obligan al programa a romper este orden y saltar a otra instrucción que no es la siguiente.*

Por ejemplo GOTO 80 *hace que el programa, al leer esta instrucción vaya directamente a la instrucción número* 80, *esté donde esté en este momento.*



IL31

> *Los programas se ejecutan secuencialmente desde la línea de programa de número más pequeño a la de número más grande, excepto que alguna línea le obligue a saltar a otra línea lejana.*

Vamos a ver un caso práctico:

 (F2) (RETURN) *(llamamos a* AUTO*)*

 KEY OFF: CLS (RETURN) *(Borramos línea de teclas de función y la pantalla)*

 nu = nu + 1 (RETURN) *(*nu *vale lo que valía más 1)*

 LOCATE 15,12: ? nu (RETURN) *(Imprime el valor de* nu*)*

GO TO 20 (RETURN) *(ir a 20 otra vez y repetir)*

(CTRL) + (STOP) *(terminar la numeración automática del programa)*

(F4) (RETURN) *(listar el programa)*

¿Que hará el programa?. Veamos cada una de las instrucciones.

- *La primera instrucción borra toda la pantalla.*
- *La segunda instrucción suma uno al valor de* nu *y se lo carga al propio* nu.
- *La tercera imprime* nu.
- *La última hace que el programa repita la sentencia número* 20 *volverá a sumar uno a* nu, *volverá a imprimir el nuevo resultado y así sucesivamente.*

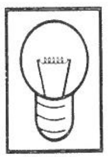
  Es como un ciclo de operaciones que se repiten.

Ahora vamos a ejecutar el programa y verás que irá imprimiendo números consecutivos a gran velocidad. Cuando quieras parar, aprieta la tecla STOP.

RUN (RETURN)

¡¡Menuda velocidad!!. Como ves, va imprimiendo repetidamente números consecutivos del 1 *hasta que paremos.*

(STOP)



IL32

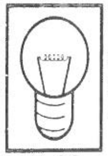
La tecla (STOP) *detiene la ejecución de un programa. Para continuar, volver a apretar* STOP.

En realidad, cuando se para un programa con STOP, *todo el teclado queda anulado y solo funcionan las teclas* (STOP) *y* (CTRL)

Si queremos que el programa prosiga

(STOP)

y el programa prosigue. Tan solo ha hecho una pausa.



IL33

Para activar y desactivar la pausa, pulsar sucesivamente (STOP)

Si, por el contrario, queremos "salir" de la ejecución por un tiempo más largo, para hacer otras cosas...

(CTRL) + (STOP)

Observa el mensaje que proporciona el MSX-C:

BREAK IN.....

que significa:

RUPTURA EN LINEA...

El MSX-C *cuando hacemos un* BREAK, *o sea* (CTRL) + (STOP) *nos informa de la última instrucción que ha ejecutado correctamente.*

Si hemos parado con CTRL + STOP, *podemos hacer ahora cualquier otra a condición de no tocar las líneas de programa, si queremos que continue donde estaba.*

En estos momentos estamos parados y nos interesa ver el programa, por ejemplo:

(F4) (RETURN)

Vemos el programa para consultar cualquier cosa. Si queremos continuar...

CONT (RETURN)

y como verás el programa continua.

Vamos a pararlo con "ruptura", "rompiéndolo" *temporalmente.*

(CTRL) + (STOP)

Ahora vamos a modificar el programa que tenemos. Haremos que a "nu" *se le vayan sumando de* 13 *en* 13, *en lugar de* 1 *en* 1. *Para esto, situate en la línea* 20, *con el cursor después del* 1 *del final (a continuación de él) y:*

3 (RETURN) (HOME)

Hemos modificado el programa y nos hemos ido al origen para...

CONT (RETURN)

El MSX-C *responde:* CAN'T CONTINUE. *Esto quiere decir:* "No puedo continuar".

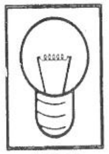
¿Porqué?

El MSX-C *lo hace así porqué se ha modificado el programa y no hay seguridad de lo que se estaba ejecutando prosiga sin errores.*

Podríamos continuar forzando al MSX-C *a ir a una determinada línea del programa, a pesar de que el* MSX-C *ha dicho que no podia continuar.*

GO TO 20 (RETURN)

Pero si observas con detenimiento, verás que el programa ha empezado con la variable nu *con valor* 0.



IL34

Cada vez que se modifica un programa, todas las variables que contenga, se ponen a cero si son numéricas y a cadena nula (" ") si son de cadena.

Por esto, si "tocamos" un programa mientras estamos en "break", *despues no obedece al* CONTINUE.

Como has visto, GOTO *también sirve para* "entrar" *en un programa por la línea que queramos.*

Si has dejado funcionando el programa, debes tener un bonito número en la pantalla.

(CTRL) + (STOP)

(SHIFT) + (HOME) new (RETURN)

FIN DE LA SESION 10

¿Que hemos aprendido?

- GO TO *significa* "IR A...."
- *Obliga al control del programa a saltar a otra línea lejana.*
- *Con* GOTO *se pueden hacer programas que tengan ciclos repetitivos.*
- (STOP) *equivale a* "párate" *(Pausa)*
- (CTRL) + (STOP) *equivale a* "párate y sal del programa un momento" *(Ruptura)*
- CONTINUE *es para anular las paradas de ruptura. Para que* CONTINUE *funcione, no se ha debido tocar el programa.*
- *Si se modifica el programa,* CONTINUE *no funciona.*
- GOTO *sirve para comenzar la ejecución por la línea que queramos.*

EJERCICIOS DE LA SESION 10

28°.- *Haz un programa que llene toda la pantalla de "H".*

29°.- *Para la ejecución y modifica el programa para que escriba* "O" en *lugar de* "H".

30°.- *Páralo otra vez. En vez de "O", y en su lugar...*

[GRAPH] + [J]

Ejecuta.

SOLUCIONES A LOS EJERCICIOS DE LA SESION 10

28°.- 10 KEY OFF: CLS

20 PRINT "H";: GOTO 20

[F5]

Como ves, GOTO *también salta al comienzo de la línea en la que está.*

29°.- [CTRL] + [STOP]

[F4] [RETURN]

[⇧] *(2 veces) y* [⇨] *(10 veces)*

O [RETURN]

[F5]

30°.- [CTRL] + [STOP] [F4] [RETURN]

[⇧] *(2 veces) y* [⇨] *(10 veces)*

[GRAPH] + [J] [RETURN]

[F5]

SESION 11

El MSX-C *tiene un reloj que arranca en el momento de encenderse. Este reloj es en realidad un contador que suma uno a una variable propia del* MSX-C *cada* 1/50 *segundo.*

Esta variable se llama TIME, *y puedes hacerla salir a la pantalla.*

SENTENCIAS	FUNCION
? TIME (RETURN)	*Imprime* "TIME"

Verás que sale un número que expresa cuantas fracciones de 1/50 *segundos hace que conectamos el* MSX-C.

Hazlo.

? TIME (RETURN)

Si queremos ver como va cambiando "TIME":

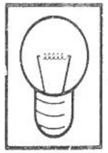
5 KEY OFF: CLS (RETURN)

10 LOCATE 15,12 : PRINT TIME
GOTO 10 (RETURN)

(F5)

La instrucción número 5 *limpia la pantalla.*

La instrucción número 10 *se situa, imprime el valor de* "time", *vuelve a situarse, vuelve a imprimir y así sucesivamente.*



IL35

Podemos hacer que "TIME" *arranque o empiece en un valor cualquiera.*

Si el programa sigue funcionando, páralo con:

CTRL + STOP

Esta combinación de teclas, "rompe", *cancela, el programa que se esté ejecutando.*

Ahora:

F4 + RETURN

7 TIME = Ø RETURN *(Reinicializamos "time" con el valor Ø).*

⇨ *(30 veces)* DEL 7 RETURN

RUN RETURN

Como ves, arranca desde cero siempre.

CTRL STOP

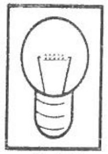
TIME *tiene un valor máximo de* 65.535. *Cuando alcanza este valor, vuelve a comenzar desde cero. Compruebalo haciendo:*

7 TIME = 65000 RETURN

⇨ *(31 veces)* 10 RETURN

RUN RETURN

Como ves, arranca desde 65000 y cuando llega a 65535 y sigue con Ø.



IL36

Si el valor de TIME *lo dividimos por* 50, *estamos controlando el tiempo en segundos.*

Si no está parado el programa....

CTRL + STOP

Veamos cuanto tiempo lleva encendido el MSX-C:

? TIME/50 RETURN

El número que sale es el tiempo, en segundos, desde la última inicialización del TIME.

?TIME/50/60 RETURN

es el tiempo, en minutos, desde que TIME tuvo por última vez el valor Ø.

Vamos a practicar con una nueva instrucción, la instrucción o sentencia IF.....THEN.

IF...THEN *significa* "SI...ENTONCES"

NEW RETURN

Esta instrucción "NEW" *limpia (borra) todo lo que haya en memoria.*

IF 2*3 = 6 then PRINT "Cierto" RETURN

Como ves se ha imprimido la palabra "cierto" *pues* 2*3 *es igual a*6.

La instrucción anterior se "leería" *así por el* MSX-C.

Si 2 por 3 es igual a 6, entonces imprime "cierto".

O sea que si lo que se expresa después del IF *es cierto, entonces se ejecuta lo que hay después del* THEN.

Otro ejemplo:

10 if 2 + 5 = 8 THEN PRINT "cierto" (RETURN)

20 PRINT "falso" (RETURN)

(F5)

Como ves, si la expresión que hay después del IF *no es cierta, no se ejecuta lo que hay después del* THEN, *sinó que se realiza la instrucción siguiente.* (la 20).

Ahora modifica el "8" *de la instrucción número* 10, *y pon un* "7" *para que la expresión después de* IF *sea correcta, y por tanto se ejecute lo que está después de* THEN.

(⇧) *(5 veces) y* (⇨) *(10 veces) (situarse sobre el* 8)

(7) (RETURN) (RETURN) (RETURN)

Ha escrito "cierto" *y también* "falso", *los dos* "PRINT"

¿Que ha pasado?

Algo importante y sencillo.

1°.- Por ser cierta la expresión 2 + 5 = 7, *ejecuta el print* "cierto" *porqué es lo que está después del* THEN.

2°.- Ahora, como ya ha terminado de ejecutar la instrucción 10, *pasa a ejecutar la siguiente* (20) *que es el print* "falso".

Comprobamos que siempre, lo que hay en la línea 20 *se imprimirá (falso), incluso cuando la operación sea correcta. Para evitar este problema, la sentencia* IF.....THEN *puede ser más potente. Así:*

IF..........THEN..........ELSE..........

Que significa:

IF	THEN	ELSE
SI (esto es cierto)	Entonces (haz esto)	Sino (haz esto)

Veamos:

NEW (RETURN)

IF 5 ^ 5 = 28 THEN PRINT "Cierto" ELSE PRINT "Falso"

(RETURN)

(F5)

Como es lógico imprime "falso" *pues* 5 *por* 5 *son* 25.

Fíjate que como la expresión después de IF *no es cierta, no se ejecuta lo que hay después de* THEN *y en cambio si que se ejecuta lo que hay después de* ELSE.

Modifiquemos la sentencia haciendo que la expresión 5 ^ 5 = 25.

(⇧) *(4 veces)* (⇨) *(8 veces)*

5 (RETURN)

¿OK?. Como ves, ahora dice "cierto" *o* "falso", *pero no las dos cosas como antes.*

Vamos a ver una aplicación muy bonita de todo esto.

Vamos a construir un reloj digital que cuente segundos. Las condiciones iniciales que nos imponemos para ello, son las siguientes:

- *Ha de comenzar en Ø*
- *Ha de permanecer permanentemente en pantalla.*
- *Se usará la variable* TIME.

Ya sabes que TIME *aumenta de* 50 *en* 50 *cada segundo. Vamos allá:*

(F2) (RETURN) *(Hacemos que el* MSX-C *numere las líneas)*

KEY OFF: CLS (RETURN) *(limpiamos la pantalla)*

IF TIME<50 THEN GO TO 20 (RETURN)

(mientras TIME *sea menor a* 50, *es decir, a un segundo el programa se quedará en esta línea. Cuando* TIME *sea igual a* 50, *es decir, que haya pasado un segundo, el* MSX-C *saltará a la línea siguiente)*

LOCATE 15,12:PRINT s (RETURN) *(imprimirá el valor s)*

S = S + 1 (RETURN) *(se suma* 1 *a* S)

TIME = Ø (RETURN) *(se inicializa el* TIME *a cero)*

GOTO 20 (RETURN) *(volvemos a la línea* 20*)*

(CTRL) + (STOP) *(terminamos el programa)*

(F5)

Como ves el programa comienza por Ø pues la primera vez casi seguro que TIME *es mayor que* 50 *y va directamente a imprimir* S *que como todavía no tiene valor cargado, es Ø.*

Se acumula un 1 *en* S, *ponemos el tiempo a Ø y volvemos a la línea* 20 *a esperar que* TIME *no sea menor que* 50. *En cuanto* TIME *es igual a* 50, *la expresión después de* IF *ya no se cumple y el* MSX-C *prosigue la ejecución en la línea* 30. *Ahora* S = 1 *y el* MSX-C *pondrá un* 1 *en la pantalla. El ciclo se repetiría indefinidamente.*

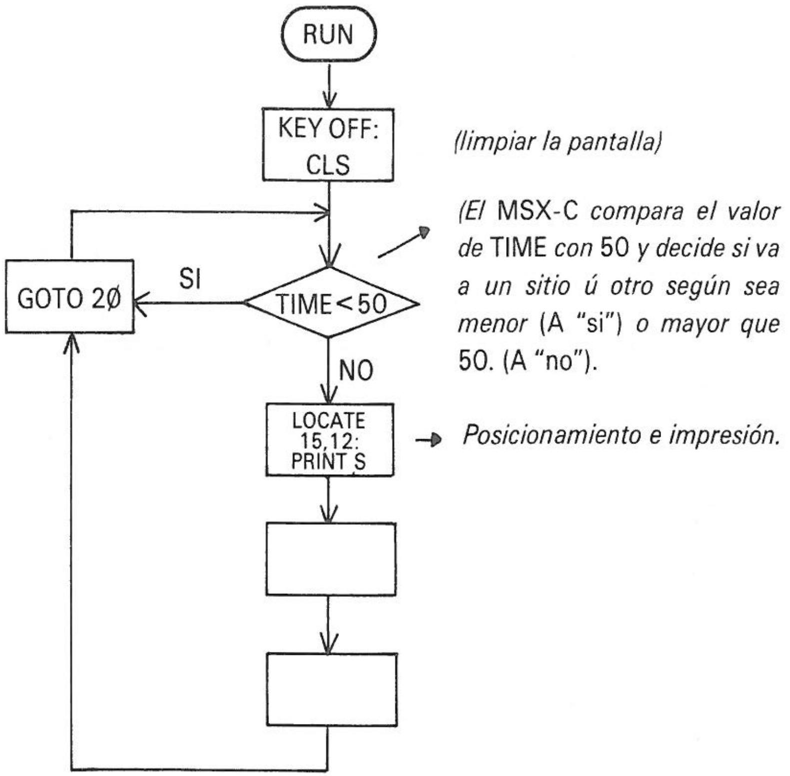
Puedes pararlo y estudiar lo que tienes en pantalla a fondo.

(CTRL) + (STOP)

LIST (RETURN)

FIN DE LA SESION 11

31°.- *Ordinograma (esquema) del programa del reloj:*



Si se sigue el orden de las flechas se comprenderá que este dibujo es un reflejo del programa que hay en pantalla.

Ejercicio: *En los dos rectángulos vacios falta algo. Se trata de ponerlo y explicar lo que es y que hace.*

a) *Falta la instrucción:*

Que suma un segundo a los que ya hubiera anteriormente.

b) *Falta la instrucción*

que inicializa el reloj interno. (Pone a cero el reloj)

SESION 12

En la presente sesión haremos un repaso de lo más importante que hemos visto, pero antes, aprenderemos algo muy importante también.

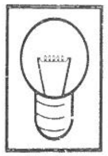
Suponte que tenemos una lista de nombres y tallas:

JUAN . . . . 1,75 metros.
PEDRO . . . . 1,60 metros.
LUIS . . . . . 1,38 metros.
ARTURO . . . 1,81 metros.

Queremos introducir estos nombres, y estaturas dentro del MSX-C *para hacer algún proceso de datos con ellos.*

Está claro que los datos de entrada (NOMBRES y ESTATURAS) *son dos grupos distintos.*

Si miramos la lista anterior y pensamos... ¿Cual es el tercer nombre?. Enseguida decimos: "Luis". *Por el orden en que están escritos,* LUIS *es el tercero, es el nombre número* 3.


IL37

> *En el* MSX-C *podemos entrar datos sucesivos (diferentes datos) a un mismo nombre de variable con la condición de dar a ese nombre un número de orden, un índice.*

Por ejemplo si la variable para guardar el nombre la llamamos NO$ *(ya sabes que el* $ *es necesario pues vamos a cargar una cadena o conjunto de caracteres alfa-numéricos), está claro que el tercer nombre sería*

NO$(3)

Ese "3" *indica que* NO$(3) *es el nombre número* 3. (LUIS).

Ahora haremos un programa que nos permita entrar cuatro nombres.

ESQUEMA DEL PROGRAMA

OBJETIVO: *Entrar cuatro nombres en la memoria del* MSX-C

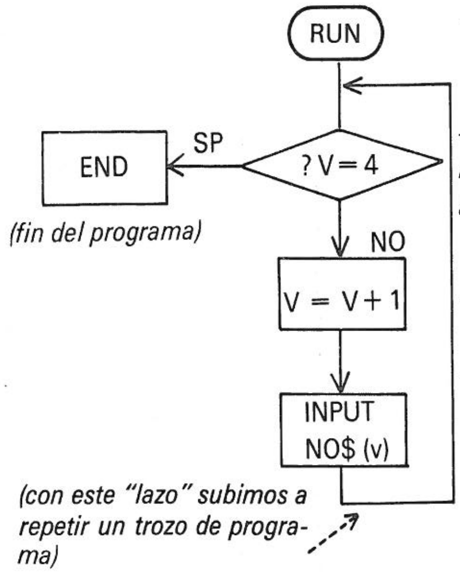
RECURSOS: *(instrucciones que vamos a utilizar para escribir el programa)*

- INPUT
- IF
- GOTO

VARIABLES: V, *veces que hemos entrado un nombre.*

NO$, variable de los nombres.

Vamos a hacer un ordinograma (un esquema) de como ha de funcionar el programa.



- *Esto significa que aquí empieza el programa.*
- *Aquí preguntamos si ya hemos entrado cuatro nombres al* MSX-C
- *Aquí se suma uno cada vez.*
- *Aquí entramos los nombres, del primero al cuarto, pues* V *va tomando los valores de* 1 a 4.

Vamos a codificar *(escribir las instrucciones del programa).*

Introduce de la manera acostumbrada, el siguiente programa. (acuerdate de que puedes utilizar AUTO*).*

```
NEW
10 IF V=4 THEN END
20 V = V + 1
30 INPUT "Entra un nombre "; NO$(V)
40 GOTO 10
```

Si has hecho servir AUTO, *acuerdate para salir de él...*

(CTRL) + (STOP)

Ahora:

(F5)

JUAN (RETURN)

PEDRO (RETURN)

LUIS (RETURN)

ARTURO (RETURN)

OK
□

Como ves, a causa del buen funcionamiento del programa, al entrar el cuarto nombre se para porqué END *es una sentencia que indica que el programa ha terminado.* END *significa* FIN *en inglés.*

Veamos el programa, paso a paso.

INSTRUCCION

[10] *El valor de* V *en este momento es todavía cero, así que el programa ignora* THEN... *y sigue en la siguiente instrucción pues* V = 4 *no es cierto.*

[20] *Ahora* V *vale* 1

[30] *Entramos* JUAN *en* NO$(1)

[40] *Salta (bifurca) a* 10

[10] *Todavia* V=4 *no es verdad pues* V=1

[20] *Ahora* V=2

[30] *Entramos el segundo nombre (Pedro) en* NO$(2)

[40] *Salta (bifurca) a* 10

[10] *Al ser* V=2, *no se cumple* V=4 *y por tanto se salta a* 20.

[20] V=3

[30] *Entramos el tercer nombre (Luis) en* NO$(3).

[40] *Salta a 10*

[10] *Salta a 20 pues no se cumple* V=4

[20] V=4

[30] *Entramos el cuarto nombre (Arturo) en* NO$(4)

[40] *Salta (bifurca) a* 10.

[10] *Se ejecuta el* END *(Fin) pues ahora si,* V=4, *es cierto.*

OK
□

Podemos ensayar si es verdad que la memoria del MSX-C *contiene los cuatro nombres.*

PRINT NO$(1) (RETURN)

En la pantalla sale Juan

PRINT NO$(3) (RETURN)

En la pantalla sale Luis.

Prueba los otros dos.

Efectivamente, en la memoria del MSX-C *han quedado emparejadas las siguientes asociaciones.*

NO$(1) = "JUAN"

NO$(2) = "PEDRO"

NO$(3) = "LUIS"

NO$(4) = "ARTURO"

Hagamos un listado del programa para verificar que está en memoria.

(F4) (RETURN)

Modifica la instrucción número 30 [30] *para que quede así:*

```
30 INPUT "Entra nombre, talla "; NO$(V),TA(V)
```

Como ves, TA *no lleva el caracterizador* $ *pues es una variable numérica. Cuando respondas al* INPUT *has de entrar por teclado una cadena (el nombre) y un número (la talla), separados por una coma.*

CLEAR (RETURN) *(borra los contenidos de las variables)*

RUN (RETURN) *(Comienzo)*

Introduce los nombres seguidos de la talla (entre el nombre y la talla, una coma) tal como figura en la lista. Si lo deseas, puedes inventarte ambos conceptos.

Cuando hayas entrado cuatro parejas de nombres y tallas, puedes...

PRINT NO$(2); "mide "; TA(2); " metros" (RETURN)

Si con ayuda del cursor vas cambiando el número 2 *por* 1, 3 *ó* 4, *irán saliendo los siguientes datos:*

JUAN mide 1,75 metros.

LUIS mide 1,38 metros.

ARTURO mide 1,81 metros

Para terminar vamos a modificar el programa para que la impresión de nombres y estaturas se realice automaticamente.

(F4) (RETURN)

```
10   IF V=4 THEN END
20   V=V+1
30   INPUT "Entra nombre, talla"; NO$(V), TA(V)
40   GOTO 10
OK
□
```

Modifiquemos [10] *para que, en vez de* END, *salte a otra línea que comience la impresión. Haz:*

```
10  IF V=4 THEN GOTO 50  (RETURN)
50  V=0  (RETURN)
60  IF V=4 THEN END  (RETURN)
70  V=V+1  (RETURN)
80  PRINT NO$(V); " mide "; TA(V); " metros"  (RETURN)
90  GOTO 60  (RETURN)
```

(CONTROL) + (STOP)

(F5)

Ya sabes que, aunque a partir de ahora no lo indiquemos, al final de una línea tienes que dar RETURN.

Como puedes ver, primero en [50] *ponemos el control de contar veces a* Ø. (V=0). *Estudia el programa completo.*

FIN DE LA SESION 12

¿Que hemos aprendido en esta sesión?

- *Con un mismo nombre, acompañado de un índice, podemos mantener en memoria un grupo, o varios grupos, de datos.*
- *Que el índice, también llamado casi siempre sub-índice, identifica uno entre todos los datos del conjunto de datos parecidos.*
- *Que para entradas repetidas de datos hemos de poner un control para terminar y ese control lo puede hacer una sentencia* IF.
- *Que un organigrama es el esquema de funcionamiento de un programa.*
- *Que* END *termina la ejecución de un programa.*
- *Que un* INPUT *puede recoger los datos de más de una variable, separados por comas.*
- *Que cuando decimos* [80] *nos estamos refiriendo a la sentencia o instrucción de la línea* 80 *del programa.*
- *Que un programa puede tener un grupo de sentencias dedicados a la entrada de datos, y que puede tener otro dedicado a la impresión de los mismos.*

EJERCICIOS DE LA SESION 12

32°.- *Haz que una vez introducidos los nombres y los metros de estatura, el programa imprima los nombres y la estatura en centímetros.*

Por ejemplo:

JUAN MIDE 175 CENTIMETROS

33°.- *Renumera el programa comenzando en el número* 1000 *la primera línea y saltando de* 500 *en* 500.

34°.- *Modificar el programa para que, además:*

- *límpie la pantalla al empezar.*
- *Imprima cada una de las salidas en el mismo sitio después de limpiar la pantalla, con una diferencia de dos segundos aproximadamente. El punto de* LOCATE *es* 7,12

32°.- *Modificar* [80] *así:*

```
80 PRINT NO$(V); "mide"; 100*TA(V); " centímetros"
```

33°.- RENUM 1000,,500 (RETURN)

verificarlo con  (F4) (RETURN)

34°.- *Una solución numerada de uno en uno:*

```
NEW
1  KEY OFF: CLS
2  IF V=4 THEN GOTO 6
3  V=V+1
4  INPUT "Entra nombre, talla"; NO$(V), TA(V)
5  GOTO 2
6  V=Ø
7  IF V=4 THEN END
8  V=V+1
9  CLS: LOCATE 0,12:PRINT STRING$(37," ")
10 TIME = 0
11 LOCATE 7,12: PRINT NO$(V); " mide "; 100 TA(V);
   " centímetros"
12 IF TIME < 100 THEN GOTO 12 ELSE GOTO 7
```

SESION 13

En la sesión anterior hemos visto como realizar varias veces un mismo grupo de instrucciones, por ejemplo para introducir datos. Para ello. usábamos el GOTO *(nº de línea) y el* IF *para controlar si ya no teníamos que repetir.*

Veamos otro ejemplo:

10 WIDTH 40: KEY OFF: CLS (RETURN)

20 IF N > 99 THEN END (RETURN)

30 PRINT USING " # # # # ";N; (RETURN)

40 N = N + 1 (RETURN)

50 GOTO 20 (RETURN)

En la instrucción 10 *hemos introducido una sentencia* WIDTH, *que en inglés significa* "ANCHO". *Con* WIDTH *podemos fijar el número de caracteres que caben en una línea. El valor más pequeño de* WIDTH *es* WIDTH 1 *(un caracter por línea) y el más grande es* WIDTH 40 *(40 caracteres por línea).*

En la instrucción 30 *introducimos el modo o forma* USING *" # # # # ".* USING *significa "utilizando", en español.*

El símbolo "#" es el símbolo de "número". Por ejemplo:

"Los pasajeros internacionales diríjanse a la puerta # 3".

Esta es una frase de un folleto de un aeropuerto que indica que los pasajeros internacionales han de pasar por la puerta número 3.

"# # # #" significa cuatro números, cuatro espacios numéricos.

PRINT USING "# # #"; 23 (RETURN)

PRINT USING "# # # # # # # # # # #"; 23 (RETURN)

¿Ves como le reserva más espacios?.

El efecto del USING "# # # #" *es que el* PRINT *coloca al número a imprimir en la derecha de una zona de tantas espacios como símbolos de sostenido* (#) *haya.*

(F5)

Como ves, hay cuatro posiciones para cada número.

(F4) (RETURN)

Ahora modifica la instrucción 30 *para que tenga* 5(#), *o sea:*

30 PRINT USING "# # # # #";N;

(RUN) (RETURN)

Como ves, ahora hay cinco espacios para los números y ocho números por línea.

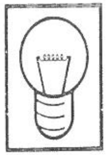
Modifica ahora la instrucción número 10 *de manera que* WIDTH 40 *sea* WIDTH 1

10 WIDTH 1: KEY OFF: CLS (RETURN)

Una vez modificada, situa el cursor en el sitio adecuado y...

(F5)

Ahora es como si estuvieras trabajando con una pantalla de un solo caracter por línea.

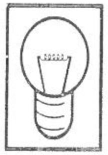


IL38

> WIDTH *determina el ancho de la pantalla (de* 1 *a* 40 *caracteres)*

(HOME)

¡Claro!. Como siempre el "HOME" *es la primera posición de la primera línea.*



IL39

> WIDTH *se encarga de centrar la pantalla de trabajo sobre la pantalla de la TV.*

(SHIFT) + (HOME)

10 WIDTH 40: KEY OFF: CLS (RETURN)

(F5)

Hemos vuelto a dar una amplitud de 40 *a cada línea de la pantalla.*

Borra ahora la instrucción [20] *tecleando.*

DELETE 20 (RETURN)

e introduce ahora una nueva sentencia:

20 FOR N = Ø TO 99 STEP 1 (RETURN)

Borra ahora [40] *y* [50]

DELETE 40 : DELETE 50 (RETURN)

y entremos una nueva instrucción [40]

40 NEXT N (RETURN)

Ahora:

[F5]

Como ves, hace lo mismo. Vamos a verlo.

[F4] [RETURN]

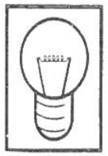
Las instrucciones [20] *y* [40] *que tienen respectivamente un* FOR... TO...STEP *y* NEXT, *definen lo que se llama un* BUCLE.

Ya deciamos en la sesión anterior que un bucle es la repetición de un número de instrucciones.

En la sesión anterior lográbamos este bucle o repetición de instrucciones, utilizando las instrucciones IF, GOTO, *y un sumador.*

FOR...TO...STEP *y* NEXT, *lo que hacen es sumar, contar y bifurcar, automaticamente.*

```
10  WIDTH 40: KEY OFF: CLS
20  FOR N = 0 TO 99 STEP1          }
30  PRINT USING "# # # #";N;       }  Esto es un bucle
40  NEXT N                         }
```



IL40

Lo que hay entre un FOR *y un* NEXT *se ejecuta repetidamente mientras la variable (en este caso* N*) tenga un valor inferior o igual al que hay después de* TO. *Cada vez que se ejecuta* NEXT, *se suma uno a la variable.* (N).

Las instrucciones FOR...TO *y* NEXT *son muy importantes en* BASIC. *A través de ellas el* MSX-C *sabe cuantas veces y en que condiciones tiene que repetir una parte del programa.*

Fíjate muy bien en su estructura:

FOR*(variable)* = *(entero)* TO*(entero 2)* STEP*(entero 3)*

GRUPO DE INSTRUCCIONES A REPETIR

FOR (variable)

variable: *Es una variable con funciones de contador que va cambiando de valor cada vez que se ejecuta el bucle. (*NEXT *le suma algo a la variable cada vez que* NEXT *se ejecuta).*

entero 1: *Valor de partida del contador.*

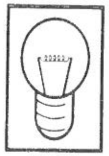
entero 2: *Valor máximo que puede alcanzar.*

entero 3: *Cada vez que se ejecuta* NEXT, *la variable con funciones de contador suma a su propio contenido esta cantidad y vuelve a* FOR *para repetir las instrucciones que están entre* FOR... *y* NEXT.

FOR *significa* "PARA"

TO *significa* "HASTA"

STEP *significa* "PASO"



IL41

Cuando la variable con funciones de sumador (también llamada variable de control) del bucle, llega a un valor mayor al máximo indicado en la sentencia FOR (TO...), *el bucle ya no se ejecuta y el programa salta y sigue en la siguiente sentencia al* NEXT.

NEW [RETURN]

[SHIFT] + [HOME] AUTO [RETURN]

Introduce ahora el siguiente programa. (Se ha omitido ☞ *y* [RETURN] *pero tu has de tenerlo en cuenta y teclearlo en cada línea).*

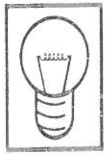
```
10   FOR A = 1 TO 30 STEP 2
20   PRINT "ESTE ES EL NUMERO"; A
30   NEXT A
```

(CTRL) + (STOP)

(F5)

Veamos que hace el programa.

El PRINT *se repite mientras el valor de* A *es menor de* 30. *Como se puede ver, empieza en* 1 *y va aumentando de dos en dos, que es el valor del paso.* (STEP).


IL42

> *Entre un* "FOR" *y un* "NEXT" *puede haber cualquier número de instrucciones.*

Vamos a aplicar esta idea para hacer un programa que entre datos de nombres y edades de un grupo de personas.

Introduce el siguiente programa usando "AUTO" *y teclea* "RETURN" *al final de cada línea:*

 (NEW) (RETURN) (AUTO) (RETURN)

```
10 WIDTH 4Ø : KEY OFF : CLS
20 LOCATE Ø,7 : PRINT STRING$(4Ø," * ")
30 LOCATE 6,10:PRINT "Entre el número de personas".
40 LOCATE Ø,14: PRINT STRING$(4Ø," * ")
50 LOCATE 17,11: INPUT PE
60 CLS
70 LOCATE Ø,5:PRINT STRING$(4Ø," * ")
80 LOCATE Ø,13:PRINT "NOMBRE......:"
```



```
90 LOCATE Ø,15:PRINT "EDAD.....:"
100 LOCATE Ø,20: PRINT STRING$(4Ø " * ")
110 FOR A = 1 TO PE
120 LOCATE 13,9: PRINT "PERSONA";A
130 LOCATE 12,13: INPUT NO$(A)
140 LOCATE 12,15: INPUT ED(A)
150 LOCATE 12,13: PRINT SPACE$(67)
160 LOCATE 12,15: PRINT SPACE$(67)
170 NEXT A
180 CLS
190 PRINT "NOMBRES
    AÑOS"
200 PRINT " = = = = = = =
    = = = = "
210 PRINT
220 FOR A = 1 TO PE
230 PRINT NO$(A);:LOCATE 36, CSRLIN: PRINT ED(A)
240 NEXT A
250 (CTRL) + (STOP)
```

El programa contiene muchas de las sentencias que hemos estudiado hasta ahora. Es verdaderamente el primer programa completo y complejo. En él se estudia fundamentalmente:

- *La sentencia* LOCATE
- *La sentencia* FOR
- *La sentencia* INPUT
- *Dos variables con subíndice*
- *La sentencia* PRINT
- *Otras sentencias y funciones*

NOTA: La ñ se consigue: [CODE] + [n]

y la Ñ: [SHIFT] + [CODE] + [N]

SPACE$(67). *Es una función que* "entrega" *al* PRINT *una cadena de 67 espacios o blancos. Es una buena función para borrar. Ya lo habíamos hecho usando* STRING$(67," "). En rea-*lidad* STRING$(67," ") *equivale a* SPACE$(67).

El acento se hace con

[SHIFT] + [`´ "]

De momento no sale nada, pero aparece al teclear la letra siguiente.

COMENTARIO DEL PROGRAMA.

1.- FUNCIONAMIENTO

[F5]

Verás que sale una pantalla que te pide que introduzcas el número de personas que quieres entrar en el proceso de introducir nombres y edades.

3 [RETURN]

Enseguida sale otra pantalla que pide el nombre y la edad de la primera persona.

JUAN GOMIS [RETURN]

23 [RETURN]

Como verás, ha cambiado el número de persona y se han borrado, solamente, los datos anteriores de la pantalla.

(tu nombre) [RETURN]

(tu edad) [RETURN]

Repite el ciclo entrando otro nombre y edad y verás que, como al comienzo le has dicho al MSX-C *que entrarías tres personas, automaticamente no pregunta más y lista en pantalla los datos introducidos.*

2.- EXPLICACION DE LAS INSTRUCCIONES

[SHIFT] + [HOME]

[F4] [RETURN]

Como ves, el programa es algo largo y se pierden de visión algunas instrucciones por arriba. Para hacer que el MSX-C *liste tan solo algunas sentencias (por ejemplo, hasta el número* 100*), se le indica así:*

[F4] -100 [RETURN]

El listado se para en la 100.

[10] *Fija el ancho de la pantalla y la borra totalmente.*

[20] *Escribe una línea de asteriscos en la línea* 7.

[30] *Escribe un mensaje de orientación.*

[40] *Escribe una línea de asteriscos en la línea* 14

[50] *El cursor se situa en el punto (columna* 17, *fila* 11*) y queda esperando el dato numérico* PE. *(número de personas)*

[60] *Una vez entrado* PE, *esta sentencia borra la pantalla.*

[70] *En la línea* 5, *otra línea de asteriscos.*

[80] *Se escribe, en la línea* 13, *un rótulo orientador.*

[90] *Se esccribe, en la línea* 15, *otro rótulo orientador.*

[100] *Se escribe otra línea, la* 20, *de asteriscos.*


(F4) 100- (RETURN)

Con esto se listan las instrucciones desde la 100 *hasta el final.*

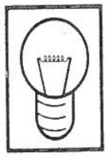
[110] *Se inicia un bucle que lo controlará el contador* A, *que como se ve, arranca en* 1 *y termina en el valor de* PE.

O sea, que este bucle se repetirá PE *veces.*

Pero... ¿que és PE*?*

Es el número de personas.

O sea, que como PE *es el número de personas y* A *empieza en* 1, *el bucle se repetirá* PE *veces. (Tres veces, en nuestro ejemplo).*



IL43

> *Cuando en una sentencia* FOR.....TO.....STEP..... *falta* "STEP", *se toma como* "STEP" *el valor* 1. *Es decir, el incremento de la variable es la unidad.*

O sea: FOR J = 5 TO 7 *equivale a* FOR J = 5 TO 7 STEP 1.

En ambos casos J *toma sucesivamente los valores* 5, 6, 7 *y* 8. *Al llegar a* 8, *como* 8 >TO7, *se salta el bucle y va a ejecutar la siguiente instrucción a* NEXT.

[120] *Escribe a que persona nos referiremos en la siguiente entrada de datos.*

[130] *Acepta el dato* NOMBRE, *en la variable* NO$

[140] *Acepta el dato* EDAD *en la variable* ED.

[150] }
[160] } *Borra estas líneas para preparar la próxima entrada.*

[170] *Incrementa en uno el control del bucle y compara con el valor del* TO...., *que en este caso es* PE.

Si el bucle ya se había ejecutado PE *veces, o sea, ya habíamos entrado la información de todas las personas, se sigue en la siguiente instrucción. Si no, vuelve a la instrucción primera del bucle, la* [120]

[180] Borra la pantalla.

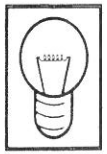
[190] }
[200] } *Ponen un título o cabecera orientativos.*

[210] *Salta una línea.*

[220] *Controla un ciclo* PE *ejecuciones.*

[230] *Imprime un nombre y una edad.*

[240] *Controla el final del ciclo y, según ya sabemos, repite* [230] *o no.*



IL44

> CSRLIN *es una función que* "entrega" *la línea en la que está el cursor.*

Es decir, CSRLIN *es una variable donde el* MSX-C *va guardando la línea donde se halla el cursor en cada momento, y su valor puede ser* "tomado" *cuando nos interese.*

De forma similar a CSRLIN, *existe otra función llamada* POS(Ø) *en la cual el* MSX-C *va guardando la posición que ocupa en cada instante dentro de la línea en que se encuentre. El formato debe ser siempre* POS(Ø).

Ejemplo:

```
10  LOCATE 12,12,1
20  A = POS(0): B = CSRLIN
30  LOCATE A,B: PRINT "PEPE"
```

(F5)

Observa como "A" *ha recogido la posición del cursor en la línea* POS(Ø) *y* "B" *la línea en la que está el cursor.* (CSRLIN).

FIN DE LA SESION 13

¿Que hemos aprendido?

- *A configurar el ancho de la pantalla.*
- *Un uso de la opción* USING.
- *Borrar líneas de programa.*
- *Ejecutar un bucle con* FOR.....TO.....NEXT
- *Imprimir ñ y Ñ*
- *Acentuar las vocales.*
- *La función SPACE$*
- *La función CSRLIN.*

EJERCICIOS DE LA SESION 13

35°.- *Haz que en el ejercicio final de la sesión, las dos pantallas de introducción de datos salgan emmarcadas por símbolos de* = *en vez de* *

36°.- *Borrar el programa anterior y hacer otro que haga lo siguiente:*

Problema:

Poner en la línea central de la pantalla los primeros 8 *múltiplos de* 5, *reservando cinco posiciones numéricas a cada uno.*

37°.- *Modificar este programa para que pregunte el número del cual haremos los cinco múltiplos, y que se ejecute una y otra vez, dejando durante* 8 *segundos el resultado en pantalla antes de preguntarnos por el número siguiente.*

38°.- *Con el anterior programa realiza, sin parar, lo siguiente:*

 (F5)

 4 (RETURN) 8 (RETURN) 81 (RETURN)

ESPERA!! *¿puedes explicar que pasa?*

SOLUCION A LOS EJERCICIOS DE LA SESION 13

35°.- *Modificar* [20] [40] [70] *y* [100] *haciendo* STRING$ (40, "*") → STRING$(40, "=")

36°.- NEW (RETURN)

AUTO (RETURN)

10 LOCATE 0,12 (RETURN)

20 FOR N = 1 TO 8 (RETURN)

30 PRINT USING " # # # # # ";5*n; (RETURN)

40 NEXT n (RETURN)

50 (CTRL) + (STOP)

(F5)

37°.- 2 KEY OFF: CLS: WIDTH 40

5 INPUT "ENTRE EL NUMERO"; NU

MODIFICAR [30] **haciendo** 5*N → NU*N

50 TIME = 0

60 IF TIME < 400 THEN 60

70 GO TO2

38°.- *El* MSX-C *tiene una memoria de teclado que permite entrar datos aunque esté procesando datos anteriores.*

SESION 14

Hoy nos sumergiremos en los modos de uso de la sentencia PRINT USING. *Recuerda que* PRINT USING *significa algo así como "escribe usando...."*

Frecuentemente desearemos que nuestros datos salgan por pantalla con un orden o formato determinado. Por ejemplo: encolumnados, con dos decimales, tal como son, solo en parte, etc.

Para todo ello existe PRINT USING.

La sentencia PRINT USING *requiere una información de control que se le proporciona con una constante de cadena que puede contener tan solo unos pocos caracteres de control. Veamos algunos ejemplos.*

(Se prescinde de  *y* (RETURN) *aunque tú ya sabes que hay que darlo en cada línea).*

```
   AUTO
10 WIDTH 40: KEY OFF: CLS
20 FOR A = 2 TO 40 STEP 2
30 PRINT USING "####";A;
40 NEXT A
   (CTRL) + (STOP)
   (F5)
```

Como verás, este programa imprime los veinte primeros números pares, reservando a cada uno 4 posiciones de impresión. WIDTH 40 *proporciona 40 posiciones de impresión por línea de pantalla, así que cada línea contiene 10 números.*

La reserva de 4 posiciones para cada número es consecuencia del control que efectúa la cadena "####" (El ; después de las cadenas de control, es obligatorio).

Vamos a verificar la cadena de control haciéndola "# # #. # #" Veamos como modificar el programa. (RETURN *en cada línea).*

(SHIFT) + (HOME)		*(limpiar pantalla)*
(F4)		*(listar programa)*
(⇧) *3 veces* (⇨) *18 veces*		*(situarse)*
(INS)	. #	*(incluir un . y un #)*
(⇨) *5 veces* (DEL)		*(borrar el ;)*
(RETURN)		
(SHIFT) + (HOME) (F5)		*(limpiar y ejecutar)*

Como ves, debido a "# # #. # #" cada número reserva tres enteros y dos decimales, y al haber desaparecido el ; después de A*, la impresión de cada número sale en una nueva línea.*

Aclarando: # # # . # #

\# *significa posición numérica*

. *significa punto decimal.*

Modifica ahora el programa para que su [20] *, STEP sea .7. O sea, deja que* [20] *sea:*

20 FOR A = 2 TO 4Ø STEP .7

Una vez modificado, ejecuta el programa con F5 *ó* RUN *y deja que termine.*

Verás que termina en 39,80 *porqué* 39,80 + .7 = 40,5*, que es mayor que el límite de* 4Ø *del* FOR.

También observarás el perfecto encolumnado que se consigue con la cadena de control del PRINT USING.

A veces desearemos que los números, cuando son grandes, salgan con las comas que acostumbran a separar los grupos de miles (cada 3 cifras). Por ejemplo:

1579413 desearemos que salga: 1,579,413

Además, puede que nos interese que, a su lado, salga algún mensaje fijo expresando el tipo de unidad que cuantificamos, o cualquier otro mensaje. Por ejemplo:

1,579,413 pesetas

Vamos a montar una cadena de control de USING *que contemple:*

- SEPARADORES DE MILES
- MENSAJES INCORPORADOS

Hagamos otro programa, pero antes borremos el anterior.

NEW

Comprueba que no hay ningún programa en memoria:

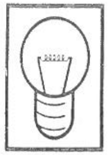
F4 RETURN

Hagamos ahora un programa que escriba separadores y mensajes anexos:

```
10 FOR A = 1 TO 15ØØØØØ STEP 12673
20 PRINT USING "# # # # # #,. # # pesetas";A
30 NEXT A
F5
```

El mensaje sale en la posición que lo has escrito. La , antes del . es el control del separador de miles.

Sin embargo, observamos algo extraño. Un símbolo % sale antes del número. ¿Que significa?



> Un % nos avisa que hemos reservado pocos espacios de impresión para el valor que estamos imprimiendo.

IL45

En este caso nos hemos olvidado de reservar espacio para las comas separadoras, así que hemos de poner 3 símbolos más de # en la cadena de control, pues tal como está, el mayor número que podríamos escribir sin problemas sería:

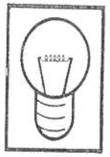
999,999.00 (7 / 3) aceptable bajo: # # # # # #, . # # (7 / 3)

Para poder escribir sin problemas

9,999,999.99 (9 / 3)

es necesaria la cadena de control

,. # # (9 / 3)



> Considerando los símbolos (#) (,) y (.) hemos de tener en cuenta que cada uno que figure en una cadena de control, representa una posición precisa de impresión.

IL46

*Si en la cadena de control, cuando se trate de controlar impresiones numéricas, substituimos los dos # # de la izquierda, por dos **, conseguiremos que los espacios libres de la izquierda de los números menores, salgan cubiertos por asteriscos.*

Modifica [20] *para que sea:*

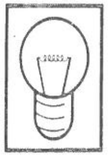
20 PRINT USING "**# # # # # #,.# # pesetas";A

Ahora ejecuta. (Cuando veas asteriscos, para con STOP).

Como ves, con ello lo que se consigue (sobre todo en listados con impresora), es tener la seguridad de que no se ha dejado ninguna cifra por escribir por un fallo de impresión. Así por ejemplo:

***12,674.ØØ pesetas

sabemos positivamente que son doce mil seiscientas setenta y cuatro pesetas, y no otra cantidad de la que no se haya imprimido alguna cifra por un fallo de la impresora, o porqué no haya tinta.



> En la cadena de control, los (**) también reservan y cuentan como dos espacios de impresión.

IL47

Para propósitos de notación científica hay también unos caracteres de control que comentaremos.

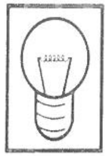
Borra el actual programa:

NEW

y entra el programa siguiente:

```
10 FOR A = 2ØØØØ TO 1 STEP -2ØØØ
20 PRINT USING "# #.# # # ^^^^";SQR(A)
30 NEXT
```

Ahora, ejecuta este programa.



> El grupo de control de 4 caracteres^(^^^^)reserva cuatro posiciones para la notación científica obligada cuando está presente en una cadena de control.

IL48

Hasta aquí el control de la impresión de números.

Vayamos ahora con el control de la impresión de valores alfanuméricos, de string o de cadena, que de todas estas maneras se puede llamar.

Con el control de impresión de contenidos de cadena podemos hacer tres cosas:

1.- Escribir el primer caracter de una cadena.

2.- Escribir los dos primeros caracteres o más que deseemos de una cadena.

3.- Escribir toda la cadena.

Vayamos por partes.

1°.- Escribir el primer caracter.

Para ello se usa la cadena de control "!"

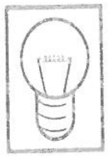
Ejemplo:

```
?USING "!";"ABRACADABRA"
```

Como ves se escribe A

```
a$ = "JOSE"
b$ = "MARIA"
PRINT USING "!";a$;b$
```

y escribe JM

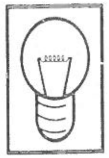


Una cadena de control puede actuar sobre las diversas variables o expresiones que esten presentes en la sentencia PRINT USING

IL49

"!" ha actuado dos veces, una sobre a$ *y otra sobre* b$.

Las únicas precauciones que han de tenerse en cuenta cuando una cadena de control actua sobre varias expresiones son las siguientes:



A) *Solo puede haber una cadena de control por cada* PRINT USING.

B) *Los caracteres de control se han de corresponder, ordenadamente, con el tipo de expresión a imprimir.*

IL50

Por ejemplo:

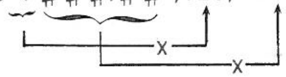
```
PRINT USING "# # #. # #!";123;"PEPE"
```



y se escribe: 123.ØØP

Pero...

```
PRINT USING "! # # #. # #";123;"PEPE"
```



Ahora escribe TYPE MISMATCH *(error en el patrón de equipo).*

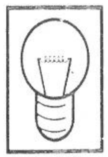
2°.- Escribir los n primeros caracteres.

Para ello se usa la cadena de control "\\".

Ejemplo:

```
?USING "\\";"ABRACADABRA"
```

y escribe AB, *que son los dos primeros caracteres.*



" \\" asignan los dos primeros caracteres de impresión, más tantos caracteres adicionales como espacios haya entre las barras invertidas.

IL51

Teniendo en cuenta que ⊔ *significa apretar una vez la barra espaciadora, haz:*

```
PRINT USING "\⊔⊔\";"ABRACADABRA"
```

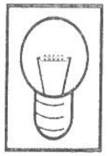
y se escribe ABRA.

Repetimos: cuando veas el símbolo ⊔ significa que has de teclear, con la barra espaciadora, tantos espacios como símbolos ⊔ haya. En la pantalla no sale nada escrito, solo se desplaza el cursor.

Veamos una cosa curiosa:

PRINT USING "!\⊔⊔\";"ABRACADABRA"

Solo escribe A.



IL52

> *Cuando en una "PRINT USING" hay mas controles de formato que expresiones a controlar, el control deja de actuar al acabarse, ordenadamente, de izquierda a derecha, las expresiones a imprimir.*

En este caso,...

PRINT USING "! \⊔⊔\";"ABRACADABRA"

habia dos patrones de formato: ! y \⊔⊔\ *pero solo habia una expresión:* "ABRACADABRA". *En consecuencia, solo actuó* ! *y solo se imprimió* A.

3.- *Escribir toda la expresión de cadena.*

Para ello se utiliza el patrón de control &.

Ejemplo:

?USING "&";a$ = "JOSE"

y escribe: JOSE. *Así de sencillo.*

FIN DE LA SESION 14

¿Que hemos aprendido?

- *Que se puede controlar el aspecto de los datos presentados en pantalla.*
- *Que en los datos numéricos podemos controlar:*
  - *Con # el número de enteros y decimales del número.*
  - *Con . la posición del punto decimal.*
  - *Con , la aparición de comas separadoras de grupo de miles.*
  - *Con ** el relleno con asteriscos de los espacios de la izquierda del número.*
  - *Con ^^^^ la obligación de representación científica.*
- *Incluir mensajes junto con los patrones de control.*
- *Que en los datos de cadena podemos controlar:*
  - *Con ! la impresión de* solo *el primer caracter.*
  - *Con \\ la impresión de los dos primeros caracteres mas tantos como espacios haya entre las barras invertidas.*
  - *Con & la impresión de* toda *la cadena.*
- *Que los diversos tipos de patrones de control que puedan existir en una cadena de control han de corresponderse con el tipo de dato que controlan.*
- *Que la cadena de control se va usando repetidamente si hay más datos a controlar que patrones de control. Hay que tener cuidado con el tipo de patrón de control y dato controlado.*

39°.- *Redacta y ejecuta un programa que, con la ayuda de la función* SQR *(raiz cuadrada), imprima encolumnadas, con cuatro decimales, las raices cuadradas de los primeros* 100 *números naturales.*

40°.- *Redactar un programa que presente, con el formato que se detalla al final, la raiz cúbica de los cinco primeros números, en orden descendente.*

FORMATO DE SALIDA: (Ejemplo de uno de los resultados)

La raiz cúbica de 3 es: 1,4422

41°.- *Redactar un programa que, a través de una* INPUT, *demande una palabra. Como resultado y en la misma línea, habrá de escribir, con 5 espacios entre si:*

- *La primera letra de la palabra*
- *Las cuatro primeras letras.*
- *Toda la palabra.*

Una vez ejecutado lo anterior, el programa pedirá una nueva entrada, hasta que esta entrada sea la letra F, *en cuyo caso imprimirá el mensaje* ADIOS.

Ejemplo de línea ejecutada:

P ␣␣␣␣␣ *PEPI* ␣␣␣␣␣ *PEPITO*

39°.-
```
1Ø FOR A = 1 TO 1ØØ
2Ø PRINT USING "##.####";SQR(A)
3Ø NEXT
```

40°.-
```
1Ø FOR A = 5 TO 1 STEP -1
2Ø PRINT "la raiz cúbica de";A;: PRINT USING "#.####";
A^(1/3)
3Ø NEXT
```

41°.-
```
1Ø INPUT A$
2Ø IF A$ = "F" THEN PRINT "ADIOS": END
3Ø PRINT USING "!␣␣␣␣␣\␣␣\␣␣␣␣␣&";A$;A$;A$
4Ø GOTO 1Ø
```

SESION 15

Profundizaremos hoy sobre aspectos ya mencionados en sesiones anteriores, incorporando de paso algunos conceptos nuevos.

Uno de estos conceptos es el de comparar cadenas de caracteres

Es fácil comparar dos números y decidir cual es mayor o si son iguales. Por ejemplo:

```
IF 3>4 THEN PRINT "4 es mayor que 3" ELSE PRINT
"3 es mayor que 4"
```

Aquí está claro que 3>4, luego, como la expresión IF *es cierta, la sentencia ejecutará la cláusula* THEN, *no la* ELSE.

4 ES MAYOR QUE 3

comparemos ahora dos letras...

```
IF A >B THEN PRINT "A mayor que B" ELSE PRINT
"B mayor que A"
```

El resultado es: "B mayor que A"

Es lógico que en este momento te preguntes: ¿Porqué B *es mayor que* A?

La razón estriba en que el MSX-C *entiende directamente los números y las letras como un conjunto de datos o entidades codificadas siempre con un valor interno.*

Por ejemplo:

```
PRINT asc ("a")
```

y se imprime el valor 97

```
PRINT asc ("A")
```

y se escribe el valor 65.

En su dia se logró un acuerdo internacional por el cual las letras mayúsculas, los números, los símbolos, y otras categorias de cosas, tuvieran

un número o valor permanente y fijo. Por ejemplo, la "a" *minúscula tendrá siempre el valor* 97.

Estos códigos fueron fijados en EEUU, *por un comité que estableció el* American Standard Code for Information Interchange *(Código Americano para el intercambio de información). Por esto a este código se le llama código* ASCII.

Pues bien, como sabes, a base de fluctuaciones o cambios de estado eléctrico, el MSX-C *trabaja sólo con números.*

¿Cómo puede identificar una letra?

La solución es fácil. ¡¡Demos un número a todo!!

```
        ┌→ código estandard americano
PRINT asc ("r")
            └→ caracter "r"
```

e imprime el número 114 *que es el valor numérico que la letra erre minúscula tendrá* siempre.

Como ves, "r" *es mayor que* "a", *pues* r *tiene un código* ASCII *de* 114 *y* a *tiene un código* ASCII *igual a* 97.

A continuación tienes la tabla de códigos ASCII *de los números y letras más comunes.*

CARACTER	CODIGO ASCII
Espacio	32
Ø	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55

8	56
9	57
A	65
B	66
C	67
D	68
E	69
F	70
G	71
H	72
I	73
J	74
K	75
L	76
M	77
N	78
O	79
P	80
Q	81
R	82
S	83
T	84
U	85
V	86
W	87
X	88
Y	89
Z	90
a	97
b	98
c	99
d	100
e	101
f	102

g	103
h	104
i	105
j	106
k	107
l	108
m	109
n	110
o	111
p	112
q	113
r	114
s	115
t	116
u	117
v	118
w	119
x	120
y	121
z	122

Hay en total, 256 *códigos* ASCII. *Aquí solo hemos representado* 63 *códigos de números y letras normales, pero hay otros códigos asignados a controles, caracteres, gráficos, alfabetos exóticos, notaciones científicas, símbolos especiales, etc.*

Si quieres saber que hace cada uno de los posibles caracteres ASCII, *hemos de dar, todavía, un paso más.*

Podemos saber que código ASCII *corresponde a un determinado caracter. Por ejemplo:*

```
PRINT asc ("M")
```

y escribe 77. (Compruébalo en la tabla).

Pero también podemos hacerlo al revés. ¿como saber a que caracter ASCII *corresponde el valor* 77?

```
?chr$ (77)
```

Imprime M, como ya sabíamos.

Si queremos conocer todos los caracteres que están asignados en la tabla ASCII, *podemos hacer un programa que los represente.*

NOTA IMPORTANTE: *Algunos códigos controlan funciones especiales tales como borrado de pantalla, brillo intenso, etc... Son los llamados caracteres de control y todos tienen código* ASCII *inferior a* 32, *así que no te extrañes si la pantalla hace cosas raras hasta el código* 32.

```
1Ø  FOR a=Ø TO 255
2Ø  LOCATE 2Ø,12 : PRIINT "EL CODIGO ASCII";a; "ES UN
    CARACTER ␣";
3Ø  IF a<32 THEN PRINT "DE CONTROL" ELSE PRINT
    CHR$(a)
4Ø  TIME=Ø
5Ø  IF TIME<5Ø THEN 5Ø
6Ø  CLS:NEXT
```

Con este programa desfilan todos los caracteres correspondientes a los códigos ASCII *que están introducidos en el* MSX-C

Ahora conviene que entiendas que en el MSX-C, 7 *es mayor que* 3, *no porque aritmeticamente sea así, sinó porque intencionadamente y de manera coherente con el anterior razonamiento, el código* ASCII *del* 7 *es mayor que el código* ASCII *del* 3 (55 *y* 51 *respectivamente).*

Lo mismo pasa con las letras:

- b es mayor que a (98>97)
- B es mayor que A (66>55)
- b es mayor que B (98>66)

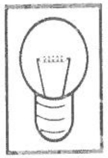
En resumen, de mayor a menor (consulta la tabla ASCII)

minúsculas	*(Del* 97 al 122)
Mayúsculas	*(Del* 65 al 90)
Números	*(Del* 48 al 57)

Ejemplos:

IF 56>123 THEN ? "56 MAYOR QUE 123" ELSE ?
"123 MAYOR QUE 56"

Imprime: 123 *mayor que* 56.



IL53

En comparación de números, siempre es mayor el que tenga más caracteres enteros significativos. Siempre se compara de izquierda a la derecha.

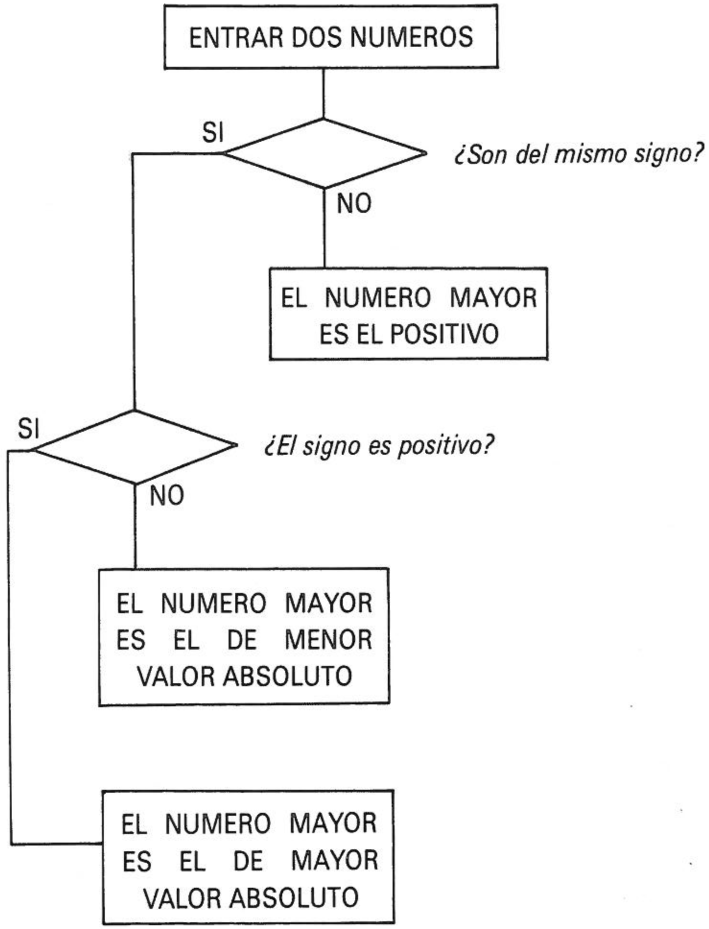
Esta regla solo se invierte cuando algún número, o los dos, son negativos.

Ejemplo:

IF 56>-123 THEN ? "56 ES MAYOR QUE -123" ELSE ?
"-123 ES MAYOR QUE 56"

y escribe... 56 es mayor que -123, a pesar que -123 tiene más cifras (pero son negativos).

A continuación vemos un esquema de como se comparan dos números.



Las cadenas de caracteres se comparan caracter a caracter, desde la izquierda a la derecha. Es mayor aquella que en un momento dado tenga un caracter más, o más alto que la otra cadena.

Ejemplo:

? "abcd">"abc"

Escribe: -1

> *Una operación de relación escribe un -1 si la relación es cierta, y un Ø si es falsa.*

IL54

```
? "abcD" >"abcd"
```

Escribe Ø, falsa. Ya que D mayúscula es menor que la d minúscula.

```
? "␣␣␣abcd">"abcd␣␣␣"
```

Escribe Ø, falsa, pues ␣ *es menor que* a.

```
? "ubaldo">"guillermo"
```

Escribe -1, cierta. Como la primera letra u *es mayor que* g, *ya no se compara más.*

```
? "Guillermo"  "guillermo"
```

Escribe Ø, falsa, porqué G *es menor que* g.

Con esto ya sabes lo principal para entender como podemos hacer una clasificación de nombres por orden alfabético.

Después de introducir cinco nombres a través de un INPUT *sobre una variable subindicada, vamos a clasificarlos de menor a mayor y obtendremos la práctica suficiente como para listar alfabeticamente cualquier secuencia de cadena.*

Primero establezcamos lo que hemos de hacer. Programemos la carga de los nombres:

```
1Ø  FOR A=1 TO 5
2Ø  PRINT "ENTRE EL NOMBRE NUMERO";A;
3Ø  INPUT no$ (A)
4Ø  NEXT A
```


Ahora programemos la clasificación de los nombres, haciendo lo siguiente:

- *Comparamos el primer nombre con los siguientes, intercambiando las posiciones si aquel es mayor. Así hasta el final.*

  Después hacemos lo mismo con el segundo, el tercero, etc.

  Cuando comparemos el cuarto con el quinto, habremos terminado.

```
5Ø  FOR A = 1 TO 4
6Ø  FOR B = A + 1 TO 5
7Ø  IF no$(A)>no$(B) THEN SWAP no$(A), no$(B)
8Ø  NEXT B : NEXT A
```

Ahora los imprimiremos.

```
9Ø  FOR A = 1 TO 5
1ØØ PRINT no$ (A)
11Ø NEXT A
```

Entra cinco nombres y comprueba el funcionamiento del programa.

Hemos de hacer aquí dos consideraciones importantes.

- *El* MSX-C *siempre dispone automaticamente de once "casillas" para cualquier variable subindicada.*

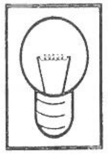
  Por ejemplo, basta emplear en un programa la variable no$(3) *para què el* MSX-C *reserve sitio para los elementos (once) del* no$(0) *al* no$(10).

- *Ahora bien, si sin previo aviso queremos usar el elemento* no$(23), *por ejemplo, el* MSX-C *no imprime nada si es en salida, como* PRINT no$(23). *Si es en una entrada, por ejemplo* INPUT no$(23), *daria error pues no tiene memoria para más allá de* no$(10).

  ¿Como hacer para reservar memoria a una tabla de elementos o variable subindicada?.

Respuesta: A través de la sentencia DIM.

Con DIM no$(1ØØ) *estamos reservando* 101 *"celdas" de memoria (de la Ø a la* 1ØØ*) para hasta* 1ØØ *nombres.*


IL55

> *Cuando una variable subindicada ha de tener más de once elementos o bien el subíndice ha de ser mayor de* 1Ø, *es necesario inicializarla (reservar espacio) previamente con una sentencia* DIM.

En la sentencia DIM, *se expresa el máximo valor que puede asumir el subíndice.*

Cuando queremos intercambiar el valor o contenido de dos variables se puede hacer facilmente con la sentencia SWAP *(intercambiar).*

Ejemplo:

a$ = "Si hoy es martes"

b$ = "esto es Bélgica"

PRINT a$; "␣"; b$

Escribe: Si hoy es martes esto es Bélgica.

Ahora hagamos:

SWAP a$, b$

y volvamos a ejecutar la misma PRINT

PRINT a$; "␣"; b$

Ahora a$ y b$ han cambiado sus contenidos, pues se escribe:

esto es Bélgica si hoy es martes

FIN DE LA SESION 15

¿Que hemos aprendido?

- *Significado del código* ASCII
- *Las comparaciones de números y letras.*
- *El orden: minúsculas, mayúsculas, números.*
- *Comparar palabras y frases*
- *Clasificar palabras*
- *La función de la sentencia* DIM
- *La función de la sentencia* SWAP.

EJERCICIOS DE LA SESION 15

42°.- *Modificar el programa que clasificaba nombres de tal manera que no procese tan solo cinco nombres, sinó que procese tantos como le informemos a través de una* INPUT *inicial. (Hasta un máximo de* 20).

43°.- *Ejecuta el programa anterior con frases en vez de nombres. Por razones que se explicarán en otra sesión, las frases no han de contener comas.*

Introduce frases como:

El perro come demasiado.

Adios muy buenas

...

...

Cuando el programa haya ejecutado la impresión, reflexiona sobre el resultado y el porqué de un orden tal como el que te saldrá.

SOLUCION A LOS EJERCICIOS DE LA SESION 15

42°.-

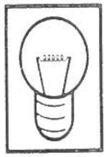
```
  2 WIDTH 40 : KEY OFF : CLS
  5 DIM NO$(20)
  7 INPUT "CUANTOS NOMBRES" ; N : CLS
 10 FOR A = 1 TO N
 20 PRINT "ENTRE EL NOMBRE NUMERO";A;
 30 INPUT NO$(A)
 40 NEXT A
 50 FOR A = 1 TO N -1
 60 FOR B = A + 1 TO N
 70 IF NO$(A)>NO$(B) THEN SWAP NO$(A), NO$(B)
 80 NEXT B : NEXT A
 90 FOR A = 1 TO N
100 PRINT NO$ (A)
110 NEXT
```

Recuerda las reglas de clasificación de números, minúsculas, mayúsculas y palabras.

SESION 16

En muchos programas hay alguna parte que debe repetirse varias veces, con lo cual se escriben una vez y se llaman cada vez que se precisan.

Otras veces, para hacer más clara la lectura del programa, éste se estructura en módulos que hacen una cosa específica dentro del programa.

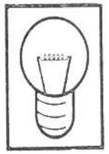


IL56

Se llama subrutina al conjunto de sentencias que, dentro de un programa, tienen un propósito particular y que son llamadas como tales por otras sentencias del mismo.

El comienzo de una subrutina no precisa ninguna instrucción especial, aunque suele ir encabezada por un comentario.

Sin embargo, es obligatorio que una subrutina termine con la sentencia RETURN, *que devuelve el control del programa a la siguiente instrucción a la que la llamó.*



IL57

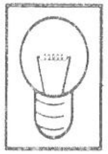
¿Que es lo fundamental de una subrutina?. Que, una vez ejecutada, el programa sigue a continuación de donde estaba cuando llamó a la subrutina.

Ejemplo:

```
10 SENTENCIA
20 SENTENCIA
30 HACER SUBRUTINA 3000
40 SENTENCIA
50 SENTENCIA
60 FIN
3000 SUBRUTINA
3001 FIN SUBRUTINA
```

como puede comprobarse, la secuencia de ejecución es:

10 → 20 → 30 → 3000 → 3001 → 40 → 50 → 60.



IL58

> *En los programas se pueden poner avisos, comentarios, líneas en blanco, cenefas, etc., para ayudar a la lectura visual de los mismos.*

Para hacer lo anterior, se usa la sentencia no ejecutable REM.

esta sentencia es, en realidad, una nota de atención para el lector humano del programa.

El MSX-C *no se queja de su presencia, pero la ignora totalmente a la hora de ejecutar el programa.*

Parecido en su función a REM, *la aparición de un acento solitario (') facilita también con gran versatilidad la oportunidad de escribir un comentario.*

Veamos un ejemplo de lo dicho hasta ahora.

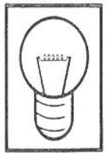
COMIENZO

```
1Ø CLS: WIDTH 4Ø: KEY OFF: INPUT "CUANTOS ELEMEN-
   TOS"; N'INICIALIZA LOS ELEMENTOS.
2Ø DIM TA $ (N)'INICIALIZA LA TABLA CON N ELEMENTOS,
   MAS EL Ø.
3Ø FOR A = 1 TO N
4Ø INPUT TA$(A)'TOMA LOS ELEMENTOS DE UNO EN UNO.
5Ø NEXT
6Ø GOSUB 1ØØØ'LLAMA A LA SUBRUTINA QUE EMPIEZA
   EN 1ØØØ.
7Ø GOSUB 2ØØØ'LLAMA A LA SUBRUTINA QUE EMPIEZA
   EN 2ØØØ.
8Ø ERASE TA$ : GO TO 1Ø'BORRA LA TABLA Y VUELVE A
   EMPEZAR
1ØØØ REM                  SENTENCIA NO EJECUTABLE
1Ø1Ø REM *************************
1Ø2Ø REM RUTINA PARA LA IMPRESION
1Ø3Ø REM ************************
1Ø4Ø REM                  SENTENCIA NO EJECUTABLE
1Ø5Ø CLS: LOCATE Ø,12: FOR A = 1 TO N: PRINT USING
     "\⊔⊔⊔⊔⊔⊔⊔\";TA$(A);:NEXT'IMPRIME LOS ELE-
     MENTOS DE LA TABLA.
1Ø6Ø RETURN'DEVUELVE EL CONTROL A 7Ø
2ØØØ'
2Ø1Ø' **************************
2Ø2Ø' RUTINA PARA EL PANTALLAZO
2Ø3Ø' **************************
```

```
2Ø4Ø'
2Ø5Ø A = TIME'TOMA LA HORA
2Ø6Ø IF TIME<A+200 THEN 2Ø6Ø'ESPERA 4 SEGUNDOS
2Ø7Ø RETURN'DEVUELVE EL CONTROL A 8Ø.
```

En este programa hay incluso un exceso de comentarios, lo cual también resulta antiestético.

Un criterio restrictivo y clarificador ha de marcar la pauta en el uso de comentario. O sea, pocos y claros.



Aunque los comentarios no sean ejecutables, no debemos olvidar que su presencia exige un elevado consumo de memoria.

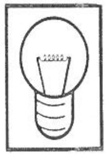
IL59

En el ejemplo anterior, tenemos dos subrutinas que son llamadas desde [6Ø] y [7Ø] *Ambas tienen propósitos diferentes.*

1ØØØ Imprime

2ØØØ Controla el tiempo que está la impresión en pantalla.

Cada una de ellas tiene una RETURN *que devuelve el control una vez ejecutada la rutina.*



El programa no ha de pasar por una sentencia RETURN *si no se ha ejecutado previamente una* GOSUB.

IL60

Esto es lógico porque si no hay un GOSUB, *el* RETURN *no sabría donde debe devolver el control.*

Compliquemos un poco las cosas.

Primero limpia la memoria:

NEW

Y ahora:

```
  1Ø' EJEMPLO DE SENTENCIA ON ... GO SUB
  2Ø WIDTH 40: KEY OFF: CLS
  3Ø INPUT "ENTRE EL VALOR DE N" ; N
  4Ø LOCATE Ø,12
  5Ø ON N GOSUB 1ØØØ, 2ØØØ, 3ØØØ
  6Ø A = TIME
  7Ø IF TIME<A + 200 THEN 70
  8Ø GOTO 2Ø
1ØØØ PRINT "EL CONTROL SALTA A LA PRIMERA DIRECCION"
     : RETURN
2ØØØ PRINT "EL CONTROL SALTA A LA SEGUNDA DIRECCION"
     : RETURN
3ØØØ PRINT "EL CONTROL SALTA A LA TERCERA DIRECCION"
     : RETURN
```

(F5)

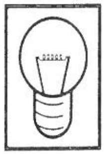
Si haces

2 (RETURN)

Verás como 5Ø *hace que se ejecute la subrutina direccionada en segundo lugar. Prueba ahora*

1

3

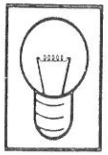


En una ON...GOSUB... *El control se cede a la dirección que figure en el lugar correspondiente al valor de la variable que está controlando el salto.*

ON (variable) GOSUB *dirección 1, 2, etc.*

IL61

Si el valor de la variable es Ø o es mayor que el número de direcciones de salto que hay, la sentencia ON...GOSUB *se ignora.*



IL62

> *También puede usarse* ON...GOTO... *pero, tal como sabemos, ya no será posible volver al punto de partida con* RETURN.

Recuerda que la diferencia entre GOSUB *y* GOTO *es precisamente el retorno o no del control.* GOTO *no maneja subrutinas, tan solo salta. (bifurca).* GOSUB *retorna a la siguiente línea después del salto, una vez ejecutada la subrutina. Este proceso lo realiza* RETURN, *por esto la* GOSUB *necesita una* RETURN *y viceversa.*

Introduce el siguiente programa:

```
     NEW
1Ø   WIDTH 40: KEY OFF: CLS
2Ø   INPUT "ENTRE UN NUMERO ENTRE 1 y 1ØØØ"; NU
3Ø   IF NU >1ØØØ THEN SA = 1 : GOTO 11Ø
     ELSE IF NU = 1ØØØ THEN SA = 6: GOTO 11Ø
4Ø   IF NU>5ØØ THEN SA = 2 : GOTO 11Ø
     ELSE IF NU = 5ØØ THEN SA = 7 : GO TO 11Ø
5Ø   IF NU >1Ø THEN SA = 3: GOTO 11Ø ELSE IF NU = 1Ø
     THEN SA = 8 : GOTO 11Ø
6Ø   IF NU>1 THEN SA = 4 : GOTO 11Ø ELSE IF NU = 1
     THEN SA = 9 : GOTO 11Ø
7Ø   SA = 5 : GO TO 11Ø
8Ø   A = TIME
9Ø   IF TIME<A + 1ØØ THEN 9Ø
```



```
1ØØ  CLS : GOTO 2Ø
11Ø  LOCATE Ø,12
12Ø  ON SA GOTO 13Ø, 14Ø, 15Ø, 16Ø, 17Ø, 18Ø, 19Ø, 2ØØ,
     21Ø.
13Ø  PRINT "NO MAS GRANDE DE 1000 POR FAVOR"
14Ø  PRINT NU; "ES MAYOR QUE 5ØØ y MENOR QUE 1ØØØ":
     GOTO 8Ø
15Ø  PRINT NU; "ES MAYOR QUE 10 Y MENOR QUE 5ØØ":
     GOTO 8Ø
16Ø  PRINT NU; "ES MAYOR QUE 1 Y MENOR QUE 1Ø: GO
     TO 8Ø
17Ø  PRINT "NO MENOR DE 1, POR FAVOR" : GOTO 8Ø
18Ø  PRINT NU; "ES IGUAL A 1000": GOTO 8Ø
19Ø  PRINT NU; "ES IGUAL A 500": GOTO 8Ø
2ØØ  PRINT NU; "ES IGUAL A 10": GOTO 8Ø
21Ø  PRINT NU; "ES IGUAL A 1": GOTO 8Ø
```

Ejecútalo!

FIN DE LA SESION 16

¿Que hemos aprendido?

- *Que es una subrutina.*
- *Que el* RETURN *devuelve el control después de una subrutina.*
- *El uso de comentarios en la línea con (')*
- *El uso de líneas de comentario con* REM *o (')*
- *La necesidad de un criterio claro y restrictivo en los comentarios.*
- *Que los comentarios ocupan memoria pero no son ejecutables.*
- *Las sentencias* ON...GOSUB *y* ON...GOTO
- *Que la variable de salto en las* ON... *puede tener cualquier valor.*
- *Que el salto solo actua si el valor de la variable de salto está comprendido entre* 1 *y tantas direcciones de salto como haya. En el caso de cualquier otro valor, se ignora. (en* ON GOSUB *o en* ON GOTO).

EJERCICIOS DE LA SESION 16

44°.- *Explica cada paso del programa anterior.*

45°.- *En una tienda puedes comprar de* 1 *a* 4 *pantalones vaqueros. El precio de un pantalón es de 1.200 ptas.*

Pero hay la siguiente oferta:

- *Por dos pantalones regalan uno.*
- *Si compras* 4 *hacen un* 40% *de descuento.*

Realiza un programa que calcule todo ello utilizando además ON...GOTO

El programa debe imprimir

PANTALONES: X

PRECIO CADA UNO: XXX

TOTAL FACTURA: XXXX

44°.- [10] *Establece* 40 *columnas*
Borra las teclas y la pantalla.

[20] *Admite cualquier número.*

[30] *Si el número es mayor que* 1000... THEN... *hace* SA = 1 y salta a [110]
Si no...
ELSE *mira si es igual a* 1000. *Si lo es...*
THEN *hace* SA = 6
Si no...
Sigue en [40]

[40] *Si el número es mayor que* 5ØØ...
THEN *hace* SA = 2 *y salta a* [110]
Si no...
ELSE *mira si es igual a* 5ØØ
Si lo es...
THEN *hace* SA = 7 *y salta a* [110]
Si no...
Sigue en [50]

[50] *Si el número es mayor que* 1Ø...
THEN... *hace* SA = 3 *y salta a* [110]
Si no...
ELSE *mira si es igual a 10. Si lo es...*
THEN *hace* SA = 8 *y salta a* [110]
Si no...
Sigue en [60]

[60] *Si el número es mayor que* 1...
THEN *hace* SA = 4 *y salta a* [110]
Si no...
ELSE *mira si es igual a* 1. *Si lo es...*
THEN *hace* SA = 9 *y salta a* 11Ø
Si no, sigue en [70]

[70] *Si llega aquí es que el número es menor que* 1, *así que automaticamente hace* SA = 5 *y salta a* [110]

[80] *Toma el tiempo*

[90] *Cuando hayan pasado* 2 *segundos* (100/50), *sigue en* [100]

[100] *Borra la pantalla y salta a* [20]

[110] *Aquí se llega desde* [30] [40] [50] [60] *ó* [70]
Coloca el cursor en la columna 0 *de la línea* 12.

[120] *Según el valor de* SA (de 1 a 9), *tomado en el grupo de instrucciones de la* [30] *a la* [70] , *salta a una de las direcciones que tiene.*

[130] *Aquí ha saltado porque* SA = 1. *Imprime y salta a* 8Ø

[140] *Aquí ha saltado porque* SA = 2. *Imprime y salta a* 8Ø

[150] *Aquí ha saltado porque* SA = 3. *Imprime y salta a* 8Ø

[160] *Aquí ha saltado porque* SA = 4. *Imprime y salta a* 8Ø

[170] *Aquí ha saltado porque* SA = 5. *Imprime y salta a* 8Ø

[180] *Aquí ha saltado porque* SA = 6. *Imprime y salta a* 8Ø

[190] *Aquí ha saltado porque* SA = 7. *Imprime y salta a* 8Ø

[200] *Aquí ha saltado porque* SA = 8. *Imprime y salta a* 8Ø

[210] *Aquí ha saltado porque* SA = 9. *Imprime y salta a* 8Ø

45°.-
```
1Ø WIDTH 40 : KEY OFF : CLS
2Ø INPUT "CUANTOS PANTALONES" ; P
3Ø ON P GOTO 5Ø, 6Ø, 7Ø, 8Ø
4Ø GOTO 2Ø
5Ø TL = 1200 : GOTO 9Ø
6Ø TL = 2400 : P = P+1 : GO TO 9Ø
```

```
7Ø  TL = 2400 : GOTO 9Ø
8Ø  TL = (1200)*P*0.6
9Ø  LOCATE 12,12 : PRINT "PANTALONES.... " ; P
1ØØ LOCATE 12 : PRINT "PRECIO DE CADA UNO = " ; TL/P
11Ø LOCATE 12 : PRINT "TOTAL A PAGAR..."; TL
12Ø A = TIME
13Ø IF TIME<A + 250 THEN 13Ø ELSE CLS: GOTO 20
```

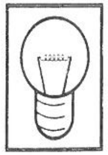

SESION 17

A veces un programa contiene en si mismo datos necesarios para su propio y correcto desarrollo.

Los datos, dentro de un programa, se situan en un tipo de sentencias no ejecutables llamadas DATA.

La sentencia REM *ó (´) no eran ejecutables. ¿Te acuerdas?. Eran datos orientativos para tí. Lo mismo pasa con* DATA, *solo que esta sentencia tiene datos para que los lea el* MSX-C.

El MSX-C *lee los datos de* DATA *con la sentencia* READ



IL63

> *En el* MSX-C *hay sentencias que tienen que ir por parejas.*

Son:

- FOR y NEXT
  Controlan bucles. A veces también está presente STEP.
- IF y THEN.
  Controlan bifurcaciones. A veces también está presente ELSE.
- GOSUB y ON....GOSUB *van con* RETURN.
- *Una* ERASE *necesita la presencia de* DIM.
- READ *siempre va con* DATA.
  A veces también está presente RESTORE.

Vamos con este último juego: READ, DATA, RESTORE. READ *lee datos sucesivos en una* DATA.

Ejemplo:

```
    AUTO
1Ø  FOR A = 1 TO 7
```

```
2∅  READ N, DI$
3∅  PRINT "El dia "; N; "semanal es el ␣ "; DI$
4∅  NEXT
5∅  DATA 1, Lunes, 2, Martes, 3, Miercoles, 4, Jueves, 5,
    Viernes, 6, Sábado, 7, Domingo
```

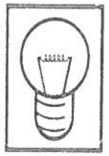
(CTRL) + (STOP)

(F5)

Como ves, la READ *ha leido parejas de números y cadenas. En la* DATA *las cadenas no tienen que ir entre comillas, a no ser que por sí mismas lleven alguna coma.*

[2∅] READ N, DI$ *quiere decir: Lee los datos, el primero será numérico y se llamará* N, *el segundo será de cadena y se llamará* DI$.

[3∅] *Imprímelos.*



Cada vez que se lee una data, un índice interno queda señalando el siguiente dato que ha de ser leido. De esta manera es automaticamente fácil incorporar más datos.

IL64

(SHIFT) + (HOME)

(F4) (RETURN)

(⇧) *(3 veces) y* (⇨) *(una vez)*

(DEL) *(borra el* ∅ *de* **[5∅]** *)* (RETURN)

Has introducido la línea 5 DATA..... *ahora, para que no esté dos veces la* DATA, *borra* **[5∅]** . *Encima de* OK...

5∅ (RETURN)

(F4) (RETURN)

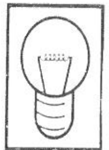
Comprueba que solo hay una DATA *y que está al comienzo del programa.*

Si te gusta más numerar las instrucciones de 1∅ *en* 1∅.

(RENUM) (RETURN)

(F4) (RETURN)

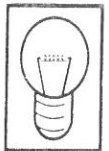
(F5)



No importa la situación de las datas en un programa. Ello es prueba de que no son sentencias ejecutables.

IL65

De todas maneras....



Si hay varias DATAS, *estén donde estén, el* MSX-C *las va leyendo una a continuación de la otra como una sola data.*

IL66

Si decimos que todas las DATA *se leen como una sola... ¿Porqué, a veces, es necesario ponerlas como grupos con distinto número de línea?*

La respuesta estriba en el uso de la nueva sentencia RESTORE, *que explicaremos a continuación.*

RESTORE *hace que el puntero que apuntaba al siguiente dato, vuelva a apuntar al primer dato de la primera* DATA, *esté el puntero donde esté.*

Primero veamos como funcionan READ *y* DATA *cuando hay más datos que lecturas efectuadas, y además los leemos en dos tandas.*

Introduce el siguiente programa.

```
     NEW
1Ø   FOR A = 1 TO 4
2Ø   READ N, DI$
3Ø   PRINT "El dia "; N; "semanal es el ␣"; DI$
4Ø   NEXT
5Ø   GOTO 10
6Ø   DATA 1,Lunes, 2, Martes, 3, Miercoles, 4, Jueves, 5,
     Viernes, 6, Sábado, 7, Domingo.
```

(F5)

Todo va bien, pero al final hay un error.

ØUT of DATA in 2Ø
(Fuera de DATA *en la línea* 20)

Veamos que hace el programa.

- [10] [20] [30] [40] , *la primera vez que son ejecutadas, leen e imprimen las cuatro primeras parejas de números y dias, pues* FOR A = 1 TO 4 *hace repetir cuatro veces esta lectura e impresión.*
- [5Ø] *vuelve a empezar el ciclo en la instrucción número* 1Ø.
- [1Ø] [2Ø] [30] [40] , *la segunda vez que son ejecutadas, intentan leer una octava pareja, la cual no existe, y el* MSX-C *lo avisa.*

Veamos como actuaría un RESTORE *situado después del primer bloque de las* 4 *primeras lecturas.*

45 RESTORE (RETURN)

Lístalo:

(F4) (RETURN)

Como ves, hay un RESTORE *después del* NEXT. *Según sabemos, es de esperar que* RESTORE "resitue" *el punto de datos al comienzo de la* DATA, *así que cuando vayamos a leer otras cuatro parejas de datos, tenga suficientes otra vez.*

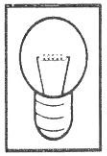
(F5)

¡¡ Tiene tantos que nunca para !!

(STOP)

Observa que cuando imprime el 4 *y el miércoles, vuelve a empezar.*

Esto no tiene utilidad práctica pero si educativa. Es evidente que cada vez que se ejecuta RESTORE, *se* "inicializa" *de nuevo la* DATA.



IL67

RESTORE *inicializa* TODO *el conjunto de* DATAS *que pueda haber en un programa, situando el puntero en el primer dato de la primera* DATA.

¿Podríamos inicializar una sola DATA, *aunque hubiera más?*

Sí. Partiendo de la base de que cada sentencia DATA *tiene un número de línea.*

RESTORE *(nº de línea)*

Sitúa el puntero al principio de la DATA *con ese concreto número de línea.*

"Rompe" *el programa:*

CTRL + STOP

NEW RETURN

Introduce el siguiente programa:

```
1Ø  WIDTH 4Ø : KEY OFF : CLS
2Ø  FOR V = 1 TO 2
3Ø  FOR A = 1 TO 6
4Ø  READ B
5Ø  PRINT USING "# # # #"; B;
6Ø  NEXT A
7Ø  RESTORE 11Ø
8Ø  NEXT V
9Ø  DATA 11, 12, 13, 14
1ØØ DATA 21, 22, 23, 24
11Ø DATA 31, 32, 33, 34
12Ø DATA 41, 42, 43, 44
```

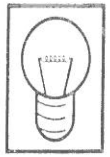
CTRL + STOP

F5

Como ves, después de haber leido 6 *datos debido al* RESTORE 11Ø, [70] *se situa en* 11Ø *el puntero y debido a* [80] *ejecuta otras 6 lecturas pero no sigue donde estaba (dato* 23) *sinó al comienzo de* [11Ø] *(dato* 31).

Al terminar, el puntero apunta al dato 43 *en la línea* [120]

Como última precisión, ha de tenerse en cuenta lo siguiente.



IL68

> *Cuando desde* READ *se lee una variable sobre un dato que haya en una* DATA, *el tipo de variable y de dato han de coincidir.*

Veamos que pasa cuando no es así.

Introduce:

```
    NEW
1Ø  READ A$
2Ø  PRINT A$
3Ø  READ B
4Ø  PRINT B
5Ø  DATA PEDRO,JUAN
```

F5

El MSX-C *dice:*

"Syntax error in 5Ø"
"Error de sintaxis en 50)

Claro. Juan no es numérico y B *sí.*

FIN DE LA SESION 17

¿Qué hemos aprendido?

- *Que un programa puede contener, dentro de si mismo, datos.*
- *Que esos datos se situan en sentencias* DATA.
- *Que la sentencia* READ *lee estos datos.*
- *Que cuando lee una variable, su tipo y el del dato han de coincidir.*
- *Que el índice interno, mientras no hay* RESTORE, *avanza de dato en dato leyendo consecutivamente las* DATAS.
- *Que no importa la situación de las* DATAS *en el programa pero sí la situación relativa entre ellas.*
- *Que todas las* DATAS, *en principio, se toman como un conjunto seguido de datos.*
- *Que* RESTORE *inicializa el puntero sobre el primer dato de todos.*
- *Que* RESTORE *(nº de línea) inicializa el puntero sobre el primer dato de la* DATA *con ese nº de línea.*

EJERCICIOS DE LA SESION 17

46º.- *Introducir dos* DATA *tales como:*

```
500 DATA 3, 7, 8, -15
600 DATA 8, 9, -3, 0
```

Hacer un programa que imprima, línea a línea, el producto de los primeros, segundos, terceros y cuartos datos de las DATA.

La salida ha de ser del tipo:

3 x 8 = 24

NOTA: *Dejar* 500 y 600 *como número de línea para las* DATA, *pero el programa iniciarlo con* [10]

47º.- *Pon en una* DATA *tu nombre, edad, domicilio y número de teléfono. Haz un programa que, después de haberla leido escriba:*

NOMBRE XXXX

EDAD ....

DOMICILIO ....

TELEFONO ....

SOLUCIONES A LOS EJERCICIOS DE LA SESION 17

46°.-

```
1Ø  N = N + 1
2Ø  IF N = 5 THEN END
3Ø  RESTORE 5ØØ
4Ø  FOR A = 1 TO N
5Ø  READ B
6Ø  NEXT
7Ø  RESTORE 6ØØ
8Ø  FOR A = 1 TO N
9Ø  READ C
1ØØ NEXT
11Ø PRINT USING "# # #   x # #= # # #";B;C;B*C
12Ø GOTO 1Ø
5ØØ DATA 3, 7, 8, -15
6ØØ DATA 8, 9, -3, Ø
```

Comentarios importantes:

1Ø *Es un contador de las veces que ya se llevan leidas,* + 1.

[4Ø] [5Ø] [6Ø]

[8Ø] [9Ø] [1ØØ]

Repiten la lectura y solo guardan la última significativa según N + 1

47°.-

```
1Ø WIDTH 40: KEY OFF : CLS
2Ø READ NO$, ED, DO$, TE$
3Ø PRINT "NOMBRE.........:␣";NO$
4Ø PRINT "EDAD.....: " ; ED
```



```
5Ø PRINT "DOMICILIO.....: ␣"; DO$
6Ø PRINT "TELEFONO.....: ␣"; TE$
7Ø DATA GUILLERMO RAMON RAMOS, 36, "CARDENAL
   REIG,␣31", 389␣57␣67
```

SESION 18

En esta sesión vamos a introducirnos en uno de los aspectos más atractivos del MSX-C*: El tratamiento de cadenas de caracteres.*

Hemos de dar por supuesto que una cadena de caracteres es un conjunto de información que bajo el aspecto de una constante, tiene forma de una secuencia entrecomillada de símbolos de todo tipo.

Es habitual asignar cada cadena de caracteres a una variable de "string" *o también llamada variable de cadena.*

Ejemplo:

V$ = "abracadabra"

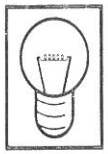
STRING *significa, desde el inglés, cadena y como cadena podemos aceptar también el significado de secuencia.*

Una variable o constante de cadena es una secuencia, cadena, de caracteres.

La máxima longitud que el MSX-C*, de manera estandard, admite en una cadena es de* 200 *caracteres.*

O sea que una variable de cadena podría ser:

a1$ = "Caliro, poeta por lo simple: [*No me importan las rimas.*
Raras veces hay dos árboles iguales, el uno junto al otro.]
Todo se resulve en ser natural y tranquilo....."



IL69

El proceso de datos no es tan solo proceso de datos numéricos, es proceso de cualquier tipo de información.

En inglés, LEFT *significa izquierda. En el* MSX-C *existe una función que se llama* LEFT *porque, operando sobre cadenas, entrega la parte izquierda de ellas.*

Veamos un ejemplo:

A$ = "Si hoy es martes, esto es Bélgica"

? LEFT $ (A$,6)

Y escribe: Si hoy

Ahora:

? LEFT $ (A$,7)

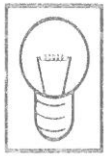
y ahora escribe: Si hoy

¿Escribe lo mismo?. ¿Que diferencia hay entre el 6 *y el* 7 *que hemos variado?*

Veamos:

? LEFT $ (A$,8)

Fíjate, ahora escribe: Si hoy e



IL70

Los espacios en blanco son caracteres como los demás, cuando forman parte de constantes o variables de cadena.

Así en nuestro ejemplo anterior, la diferencia entre A$,6 *y* A$,7 *era que en* A$,7 *aunque no se pueda comprobar visualmente, se imprimia el espacio entre* HOY *y* ES.

LEFT$ *es una función porque trata el contenido de unos datos y* "entrega" *otro conjunto de datos.*

En realidad, toda función hace el tratamiento de unos datos para entregar otros en "función" *de aquellos y del tipo de tratamiento.*

? LEFT$ ("PEPE",3)

Se escribe: PEP

Veamos la estructura de una función LEFT$

LEFT $ (CADENA, n)

número de caracteres a considerar

Puede ser una variable de cadena o una constante de cadena.

$ siempre estará presente, si lo que "entrega" *la función es una cadena.*

De la izquierda.

LEFT$ "extrae" *una subcadena de otra cadena.*

? LEFT$ ("PEPE",8)

Escribe: PEPE

Efectivamente, aunque "PEPE" *solo tiene* 4 *caracteres, si escogemos los primeros* 8 *que existan, encontraremos los cuatro que existen y cogerá cuatro espacios en blanco más.*

? RIGHT$ (A$,6)

Ahora nos escribe: élgica

RIGTH$ *entrega o extrae una subcadena de* n *caracteres, tomados desde la derecha, de otra cadena.*

Otro ejemplo:

? RIGHT$ (A$,26)

Escribe: es martes, esto es Bélgica

Resumen:

A$ = "Si hoy es mar tes, esto es Bélgica"

LEFT $ (A$,13) RIGHT $ (A$,15)

Veamos ahora otra función, la MID$

? MID$ (A$,6,11)

Escribe: y es martes

Como puedes comprobar ahora, MID$ *ha extraido una subcadena de* 11 *caracteres desde la posición de carácter número* 6, *tomada desde la derecha.*

Veamos el detalle de la estructura de MID$

MID $ (CADENA, COMIENZO, EXTENSION)

- EXTENSION: *Longitud de la subcadena de resultado.*
- COMIENZO: *Primer caracter de la subcadena, según su posición en la cadena principal.*
- CADENA: *Cadena que puede ser una variable o constante de cadena.*
- $: *Obligatorio, pues su resultado será una cadena.*
- MID: *significa* "Del medio"

Otra función sencilla, pero que entrega un valor numérico, no de cadena, es LEN.

La función LEN "entrega" *un valor numérico que expresa el número de caracteres que tiene una cadena, sea esta una variable o constante de cadena.*

? LEN ("PEPE")

Y escribe: 4

Porque PEPE *tiene cuatro caracteres. Ahora...*

? LEN ("␣PEPE")

Escribe: 5

Porque los espacios en blanco son caracteres.

? LEN (A$)

Ahora escribe: 33

"Si hoy es martes, esto es Bélgica", *tiene* 33 *caracteres.*

Hagamos un divertido programa donde aplicaremos lo aprendido en esta sesión.

```
    AUTO
1Ø  WIDTH 40: KEY OFF : CLS
15  A$ = "Si hoy es martes, esto es Bélgica"
2Ø  FOR A = 1 TO LEN (A$)
3Ø  LOCATE Ø,12 : PRINT MID$ (A$,1,A)
5Ø  B = TIME
6Ø  IF TIME< B + 5 THEN 6Ø
7Ø  NEXT
```

(CTRL) + (STOP)

(F5)

Como ves, el programa escribe la frase tal como lo haría una rápida mecanógrafa. Dejo a tu consideración el estudio profundo del programa.

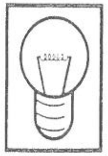
En los tratamientos de cadena, cuando hay impresión, suele resultar molesto la persistente comunicación que efectúa el MSX-C *de que ha acabado de ejecutar un programa.*

Nos estamos refiriendo a la operación del mensaje

OK

y del cursor.

Existe un truco para que ambas cosas no aparezcan. No es una solución muy elegante y en la próxima sesión atacaremos este problema de manera más profesional. De momento, ahí va una solución que no siempre será despreciable.



IL71

> *Para evitar la representación del* OK *y el* cursor, *podemos dejar el programa, en un bucle indefinido después de acabar su tarea.*

¿Como?

Veamos. En el MSX-C *todavía está el último programa que has escrito. Lístalo con:*

(F4) (RETURN)

Ahora:

```
80 GO TO 80
```

(F5)

¡¡ No aparece el OK *ni el cursor !!*

La razón estriba en que el MSX-C *está ejecutando indefinidamente la instrucción* 8Ø.

(CTRL) + (STOP)

Escribe: Break in 8Ø

Ahora si que ha parado.

Vayamos ahora a conocer una función algo más complicada, la función INSTR

Esta función nos indica la posición en la cual, dentro de una cadena, aparece por primera vez una subrutina de la misma. Veamos un ejemplo:

A$ = "Si hoy es martes, esto es Bélgica"

? INSTR (A$,"oy")

Escribe: 5

Ello es porque la subcadena "oy" *aparece a partir del carácter* 5 *de la cadena* A$

? INSTR (A$ "PEPE")

escribe: Ø

Porque no hay "PEPE" *dentro de* A$

INSTR *comienza a comparar la subcadena de búsqueda desde la primera posición de la cadena principal.*

A veces, este procedimiento es lento y, en virtud de que sabemos algo al respecto, podemos poner otra condición en INSTR.

¿Cual?

Pues que busque la subcadena desde una posición más avanzada.

? INSTR (7,A$,"oy")

Escribe: Ø

¡¡ Claro !!. Porque desde la séptima posición ya no hay ninguna subcadena "oy".

? INSTR (4,A$,"oy")

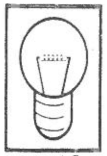
Escribe: 5. *¡¡Aquí está!!*

A continuación veremos dos funciones de uso especial que también tienen que ver con cadenas.

La función VAL *devuelve el valor de una cadena compuesta por caracteres numéricos.*

? VAL ("15400" + "7")

Escribe: 154007

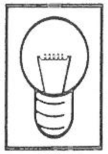


Cuando VAL *actua sobre varias cadenas, tan solo es válido el operador de concatenación.*

IL72

? VAL ("-15400" + MID$ ("12345678",5,3))

Y por tanto, nos escribirá: -15400567



Ha de tenerse en cuenta que lo que entrega VAL *es ya un valor numérico operable aritmeticamente.*

IL73

La función inversa de VAL *es* STR$. *Observa que lleva el* $ *porque........*

STR$ "entrega" *con formato de cadena,* STRING, *un número o contenido de una variable numérica.*

? STR$ (123.4)

Como prueba de que entrega una cadena, vamos a efectuar una operación de cadena sobre un valor que haya entregado una STR$

? MID$ (STR$ (123.4), 2,3)

*Escribe...*123

Sin embargo, y fijándonos bien, como hemos pedido que a partir de la posición 2 *nos escriba* 3 *números, esperábamos que escribiera:*

23

¿Porqué nos ha escrito 123?

Pues ha pasado que, aunque nosotros no lo escribamos, a todos los números el MSX-C *les reserva un espacio por delante para el signo. En los números positivos, este espacio está delante, aunque normalmente no nos fijemos en él.*

Veamos la prueba de lo anterior:

123,4 *parece tener* 5 *caracteres, pero...*

? LEN (STR$ (123.4))

Y escribe 6, *pues tiene en cuenta el espacio en blanco destinado al signo* +.

FIN DE LA SESION 18ª

¿Que hemos aprendido?

- *Una constante de cadena o variable de cadena puede llegar a tener una longitud de* 2ØØ *caracteres como máximo.*
- *Las cadenas, o informaciones literales, se pueden procesar por medio de funciones específicas.*
- *Diversas funciones de tratamiento de cadenas tales como:*

  "extraer desde la izquierda" (LEFT$)

  "extraer desde la derecha" (RIGHT$)

  "extraer desde el medio" (MID$)

  "Buscar una cadena" (INSTR)

  "Valorar una cadena de números" (VAL)

  "Encadenar un número" (STR$)
- *Bloquear la aparición del mensaje* OK *y del cursor.*
- *Que en la longitud de un número* "encadenado" *hay que tener en cuenta el espacio del signo.*

EJERCICIOS DE LA SESION 18

48°.- *Con ayuda de los códigos* ASCII *construye una cadena que contenga consecutivamente todas las letras mayúsculas* (La A es el CHR$(65) y la Z es el CHR$(90)).

Imprime, línea por línea, la cadena resultante, la longitud de la misma y la posición que ocupa la letra M.

49°.- *Redacta un programa que compruebe si algún mes del año tiene más de ocho letras y más de dos* "e" ó "E" *en su nombre.*

En el primer caso que esta situación se produzca, la pantalla debe borrarse, el nombre del mes en cuestión debe escribirse en la columna 16 *de la línea* 12, *y el programa se parará sin imprimir* "OK" *ni el cursor.*

NOTA: *Usese* DATA *para guardar la tabla de los meses.*

SOLUCIONES A LOS EJERCICIOS DE LA SESION 18°

48°.-

```
NEW
10 WIDTH 40: KEY OFF : CLS
20 FOR A = 65 TO 90
30 A$ = A$ + CHR$(A)
40 NEXT
50 PRINT A$: PRINT LEN (A$): PRINT INSTR (A$, "M")
```

49°.-

```
NEW
10 WIDTH 40 : KEY OFF : CLS
20 FOR A = 1 TO 12
30 READ ME$
40 IF LEN (ME$)<= 8 THEN 100
50 E = 0
60 FOR B = 1 TO LEN (ME$)
70 IF MID$ (ME$,B,1) = "e" OR MID$ (ME$,B,1) = "E"
   THEN E = E + 1
80 NEXT B
90 IF E>2 THEN CLS : LOCATE 16,12 : PRINT ME$ : GOTO 110
100 NEXT A
110 GOTO 110
120 DATA Enero, Febrero, Marzo, Abril, Mayo, Junio, Julio,
    Agosto, Septiembre, Octubre, Noviembre, Diciembre.
```

SESION 19

Estamos llegando al final de nuestra primera etapa.

Las sesiones que faltan están dedicados a aspectos curiosos del MSX-C.

Algunas cosas que veremos son nuevos funcionamientos de sentencias que conocemos. Por ejemplo LOCATE.

Habiamos visto que

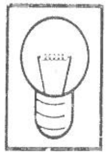
LOCATE *columna, fila*

servía para preparar una impresión en un sitio determinado de la pantalla. También sirve para controlar el lugar de una entrada de datos.

Normalmente, durante la ejecución de un programa, el cursor no aparece. El cursor puede aparecer si ponemos un 1 *como tercer parámetro de un* LOCATE.

```
1Ø   LOCATE 12,10,1
2Ø   GOTO 2Ø
```

(F5)



IL74

> *Introduciendo un* 1 *como tercer parámetro de* LOCATE, *el cursor será visible durante la ejecución de un programa, hasta la aparición de otro* LOCATE *cuyo tercer parámetro sea* Ø.

Ejemplo:

```
     NEW
1Ø   WIDTH 40: KEY OFF : CLS
2Ø   LOCATE 12,10,1
3Ø   FOR B = 1 TO 2
```

```
40   FOR A = 1 TO 500
50   NEXT A
60   LOCATE,,0
70   NEXT B
```

(F5)

Como ves, primero aparece el cursor y después desaparece (debido a LOCATE,,0) *durante un tiempo, hasta el fin del programa.*

De paso observa que en un breve tiempo, el MSX-C, *ha ejecutado* 1009 *instrucciones. Sería interesante que tú mismo investigaras cuantas instrucciones se han efectuado. Ahora...*

NEW

Veamos otra aplicación interesante.

Ya sabes que los códigos ASCII *pueden controlar, con la función* CHR$, *la aparición de caracteres normales, especiales y de control. Por ejemplo:*

LOCATE 20,12 : PRINT CHR$ (215)

El carácter ASCII 215 *es una bandera de cuadros.*

Redacta y ejecuta el siguiente programa:

```
10   WIDTH 40: KEY OFF: CLS
20   LOCATE 40*RND(B),23*RND(45),1
30   PRINT CHR$(215);
40   GOTO 20
```

Ejecútalo.

Como ves, se colocan en la pantalla una serie de banderas situadas al azar. El cursor también aparece allí donde va a ser "depositada" *una bandera. Para quitar el cursor durante la ejecución, para el programa con*

(CTRL) + (STOP)

limpia la pantalla:

(SHIFT) + (HOME)

Lista el programa:

(F4)

y borra en [20] *el* 1.

(⇧) *(4 veces)*

(⇨) *(31 veces)*

0 (RETURN)

(SHIFT) + (HOME)

(F5)

Cuando tengas suficiente...

(CTRL) + (STOP)

Puedes jugar con códigos ASCII *del* 21 *al* 254 *y ver diferentes instrucciones.*

Veamos ahora la función INKEY$

INKEY$ *es una función que entrega el último carácter tecleado.*

```
     NEW
10   K$ = INKEY$
20   ? K$
30   GOTO 10
```

(F5)

La pantalla ha quedado absolutamente *limpia!!*

Claro. El programa está imprimiendo lo que entramos por teclado, y como no estamos tecleando nada, pues nada se ve.

n

¡¡La n *sube y sube hasta desaparecer!!*

Pulsa lo que desees y como lo desees, verás que divertido. (Si pulsas STOP, *el programa para, ya lo sabias).*

Y si pulsas CTRL + STOP, *el programa* "se rompe".

¿Como parar esta ejecución endiablada y convertir en algo útil la función INKEY$?

Muy sencillo. Hagamos que si lo que se introduce no es nada, el programa entre en un buble de espera.

```
15   IF K$ = "␣" THEN 1Ø
```

(F5)

Teclea lo que desees. Verás como aparece por pantalla.

En este momento tenemos un programa que detecta lo que entramos por teclado. A partir de [15] *podemos usar este conocimiento para controlar la ejecución de un programa.*

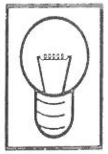
Por ejemplo, cambia la [2Ø] *haciendo:*

```
2Ø   IF K$ = "F" OR K$ = "f" THEN?
     "FIN" : END
25   ? "TODAVIA NO ES EL FIN"
```

(F5)

Teclea varias teclas y después

(F)



IL75

INKEY$ *no necesita* "RETURN". *Siempre está comprobando la última tecla pulsada.*

Para tener la precaución de que en el teclado no se haya tecleado algo inadvertidamente y esa información no controlada pueda alterar la ejecución del programa tal como lo deseamos, es conveniente "limpiar el teclado", *con un pequeño truco de programación.*

Listemos el programa:

```
5   IF INKEY$<>"      " THEN 5
```

Hay que tener en cuenta que cada vez que se ejecuta INKEY$ *se va borrando de la* "memoria del teclado" *el caracter más ultimamente tecleado.*

La sentencia que hemos introducido tiene por objeto borrar, carácter a carácter, el teclado, hasta que aquél sea inexistente o sea "   ".

```
RENUM
```

(F4)

Las [10] [20] y [30] *forman un clásico conjunto de sentencias de control del teclado. A partir de la* [30] *podemos hacer lo que queramos en consecuencia con la última tecla pulsada.*

Comprueba como se ha "Bloqueado el Teclado".

(CTRL) + (STOP)

Como última sentencia curiosa veremos que hace LINE INPUT.

Un INPUT *ya sabemos que, teniendo la posibilidad de emitir un mensaje, espera la introducción de datos de todo tipo. Tantos como se hayan definido en la sentencia. Por ejemplo:*

```
INPUT"Entre nombre y edad" ; NO$,ed
```

espera una cadena, una coma y un número. Si no le entra así, avisará del error.

Pero, ¿Qué pasa si en el dato que entramos existen comas?. Pues que INPUT *lo tomará como varias entradas. Por ejemplo:*

INPUT "Domicilio";do$

CALLE DEL PEZ, 13

y el MSX-C *responde...*

? EXTRA IGNORED *(Ignorada la entrada extra)*

Veamos:

? do$

escribe...

CALLE DEL PEZ

OK

¡Ha ignorado lo que el INPUT *(debido a la coma) supone una entrada extra.*

¿Como salir de esta trampa?

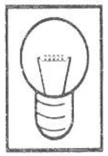
Con LINE INPUT

LINE INPUT "Domicilio ";do$

CALLE DEL PEZ, 13

? do$

Escribe: Calle del pez, 13



IL76

LINE INPUT *admite* TODA *la información que le entra y la asigna a una* UNICA *variable de cadena.*

FIN DE LA SESION 19

¿Que hemos aprendido?

- *A controlar la presencia del cursor durante la ejecución de un programa.*
- *El control de la última tecla pulsada.*
- *A dejar absolutamente limpia la pantalla.*
- *A introducir cadenas con comas como una única variable.*

EJERCICIOS DE LA SESION 19

50°.- *Haz un programa, lo más simple que sepas, que de una manera algo diferente a la explicada en esta sesión, deje la pantalla absolutamente limpia.*

51°.- *Redactar un programa que, admitiendo la introducción de texto en la línea* 22, *vaya situando, por grupos de ocho letras, ese texto en la pantalla. Debe iniciarse en la posición* (Ø,Ø), *en líneas totalmente llenas y contínuas sobre la pantalla.*

52°.- *Introducir un texto cualquiera y hacer que el* MSX-C *lo escriba en diagonal desde la posición* Ø,Ø.

SOLUCION A LOS EJERCICIOS DE LA SESION 19

50°.-

```
NEW
1Ø CLS
2Ø GOTO 2Ø
```

51°.-

```
NEW
1Ø WIDTH 4Ø: KEY OFF: CLS
2Ø GOSUB 5Ø
3Ø LOCATE A,B:PRINT TX$;:TX$ = " "
   :A = POS(Ø) : B = CSRLIN
4Ø GOTO 2Ø
5Ø K$ = INKEY$ : LOCATE POS(Ø),22 : PRINT K$;
6Ø TX$ = TX$ + K$
7Ø IF LEN (TX$) = 8 THEN 1ØØ
8Ø IF POS(Ø) = 39 THEN LOCATE Ø,22: PRINT
   STRING$ (4Ø, "␣");
9Ø GOTO 5Ø
1ØØ RETURN
```

52°.-

```
NEW
1Ø WIDTH 4Ø: KEY OFF: CLS
2Ø LINE INPUT "TEXTO ␣:";TX$
3Ø LG = LEN (TX$)
4Ø CLS
5Ø FOR A = 1 TO LG
6Ø LOCATE POS(Ø),CSRLIN + 1 : PRINT MID$ (TX$,A,1);
7Ø NEXT
```

SESION 20

En esta última sesión de instrucciones de la primera etapa del MSX-BASIC, *vamos a conocer algunas sentencias y funciones de propósito general.*

Por ejemplo:

? FRE (∅)

Te entrega un número, cuyo valor dependerá de lo ocupado que tengas el MSX-C *en este momento con datos y programas.*

Este número o valor, representa e indica la cantidad de bytes *o caracteres que aún puede admitir el* MSX-C. *Es la memoria todavía libre.*

Sin embargo:

? FRE ("   ")

Verás que se escribe un número más pequeño. Es el número de octetos libres que todavía quedan para admitir variables de STRING.

En el MSX-C *el área para variables de* STRING *se asigna con la sentencia* CLEAR.

Si no se usa una CLEAR, *que como sabes tiene también la propiedad de inicializar (poner a cero) todas las variables, el área de memoria para cadenas se inicializa con* 200 bytes *o caracteres.*

Apaga y enciende de nuevo el MSX-C.

Ahora...

? FRE ("   ")

y nos escribe... 2∅∅.

En este momento el MSX-C *está asignando* 200 *caracteres de memoria para las cadenas.*

Si esto nos parece poco y creemos que vamos a necesitar más área de STRING, *por ejemplo* 25∅∅ *caracteres, haremos.*

```
CLEAR 250Ø
```

Observemos que ha pasado:

```
? FRE ("      ")
```

que escribe... 2.500. *Tal y como esperábamos.*

> *En muchos programas que tratan variables de cadena conviene, el principio de los mismos, usar una sentencia* CLEAR *asignando área de* STRING *suficiente.*

IL77

Otra función curiosa es RND. *Esta sentencia proporciona un número decimal aleatorio entre* 0 *y* 1, *que puede servir para numerosos programas.*

```
1Ø   FOR A = 1 TO 1Ø
2Ø   PRINT RND (3)
3Ø   NEXT
```

(F5)

Como ves, se escriben 1Ø *números decimales de* 14 *cifras cada uno. Estos números tienen la propiedad de parecerse a los números aleatorios o de azar.*

(F5)

Se ha originado la misma secuencia.

Modifica ahora la instrucción nº [20] *del programa para que sea* RND(7), *y...*

(F5)

Sale el mismo tipo de secuencia!!

Y con cualquier número entero, pasaría lo mismo.

> *La función* RND *de un número positivo entrega siempre la misma secuencia de números aleatorios.*

IL78

Cambia ahora [20] *de forma que* RND(-4ØØ)

(F5)

¡¡Siempre escribe el mismo número!!

Observa también que si ejecutas RND(-34), *el número varía con relación a* RND(-400), *aunque también se repite.*

> *A los números positivos les corresponde una secuencia de números aleatorios y a los negativos les corresponde un solo número a cada uno.*

IL79

¿Como hacer pues que un programa genere números siempre distintos y "secretos"*?.*

Consiste en arrancar con un número negativo sobre el que no tengamos control y después pedir una secuencia RND *con un número positivo.*

> *En el* MSX-C *hay un número sobre el cual, si no lo deseamos, no tenemos control. Ese número es el valor de la función* TIME.

IL80

```
     NEW
1Ø   X = RND (-TIME)
2Ø   FOR A = 1 TO 1Ø
30   ? RND(5)
```

```
40 NEXT
```

(F5)

(F5)

(F5)

> *La variable* TIME, *en negativo, inicializa la tabla o secuencia de números aleatorios de manera incontrolada, lo cual es conveniente para un mayor azar.*

IL81

Viene a cuento ahora explicar otra función muy relacionada con la anterior. Esta función es INT.

```
? INT (3.8)
```

y nos escribe: 3

```
? INT 0,98
```

Escribe: 0

En el MSX-C, INT *entrega la parte entera de un número.*

Si nosotros quisiéramos números enteros aleatorios, por ejemplo entre 0 *y* 99, *no tendríamos más que multiplicar por* 100 *los números aleatorios y extraer la parte entera con* INT.

En el programa anterior...

(F4)

Modifica la instrucción [30] *para que quede:*

```
30 PRINT INT(100*RND(5))
```

Ahora, ejecuta varias veces con F5.

Observarás que salen números enteros con total apariencia de aleatorios.

> INT *es necesaria, unida a un multiplicador, para extraer números enteros aleatorios.*

IL82

En esta sesión tan variada, pasamos a explicar la sentencia que se usa en el MSX-C *para poner comentarios a los programas. Esta es* REM, *que ya la vimos anteriormente y la utilizamos para poner aclaraciones y comentarios a nuestros programas.*

A veces es bueno utilizar REM *al principio de una instrucción para convertirla momentaneamente en un comentario, y por tanto, evitando que se ejecute.*

¿Qué cuando puede ser ello útil?

Imagínate que estas probando varios trozos de un programa, y en uno de los trozos que ya has comprobado que funciona bien, hay una instrucción que te llena la pantalla de textos y te hace perder bastante tiempo. Podrías ponerle un REM *delante y así no se ejecutaría la instrucción hasta que le borraras el* REM.

> REM *es una sentencia no ejecutable, que asigna valor de comentario a todo lo que le sigue en su línea de programa.*

IL83

```
NEW
10 FOR A = 1 TO 10
20 FOR B = 1 TO 10
30 PRINT USING "##x##=####";A,B,A*B
40 S = S+A*B
50 NEXT B : NEXT A
60 PRINT S
```

(F5)

La pantalla se llena de cifras y cifras.

Supongamos que, de momento, solo nos interesa el resultado de la suma final.

¿Como evitar la impresión de las multiplicaciones sin alterar demasiado el programa?

Pues tecleando, insertando, en [30] *la palabra* REM.

```
30  REM PRINT USING"###x###=####";A,B,A*B
```

Ahora ejecutamos de nuevo el programa.

Cuando ya estamos seguros... ¿Como volver a hacer que salgan las multiplicaciones?.

Pues deleteando en [30] *la palabra* REM.

Como final y para quienes de vosotros esteis impuestos en matemáticas, hay una función que entrega en formato de cadena, el número binario de otro expresado en decimal.

```
? BIN $ (253)
```

Y escribe: 11111101, *que es* 253 *en binario.*

FIN DE LA SESION 20

¿Que hemos aprendido?

- *A observar la memoria disponible en el* MSX-C
- *A observar la memoria asignada a las cadenas.*
- *A asignar más o menos memoria en el área de* STRING.
- *A generar números aleatorios.*
- *a usar la función* TIME *para inicializar los números aleatorios.*
- *A extraer partes enteras con la función* INT.
- *La influencia del signo en la función* RND.
- *A usar comentarios con ayuda de* REM.
- *A inhibir sentencias a través de* REM.
- *A expresar un número en binario.*

EJERCICIOS DE LA SESION 20

53°.- *Redactar una sentencia que decida acabar el programa, si no tiene en el momento de ejecutarse espacio para una variable de* STRING *de* 450 *caracteres.*

54°.- *Obtener una lista de* 50 *números aleatorios, comprendidos entre* 53 y 94, *inclusives.*

55°.- Generar 10 *grupos de* 4 *letras mayúsculas al azar.*

NOTA: Ver tabla de códigos ASCII.

56°.- *Redactar un programa que presente, en binario, la tabla elemental de multiplicar.*

SOLUCION A LOS EJERCICIOS DE LA SESION 20

53°.-

```
IF FRE ("      ")<450 THEN END
```

54°.-

```
     NEW
10   X = RND (-TIME)
2Ø   FOR A = 1 TO 5Ø
3Ø   PRINT INT (53 + 42*RND (45))        (cualquier positivo)
40   NEXT
```

55°.-

```
     NEW
1Ø   X = RND (-TIME)
2Ø   FOR A = 1 TO 1Ø
3Ø   FOR B = 1 TO 4
4Ø   CLAVE$ = CLAVE$ + CHR$ (65 + 26*RND (12))
5Ø   NEXT B
6Ø   PRINT CLAVE$
7Ø   CLAVE$ = ""
8Ø   NEXT A
```

56°.-

```
     NEW
1Ø   FOR A = 1 TO 9
2Ø   FOR B = 1 TO 9
3Ø   PRINT USING "####x####=#######";VAL
     (BIN$(A)),VAL(BIN$(B)),VAL(BIN$(A*B))
4Ø   NEXT B: NEXT A
```

ANEXO 1° (Caso práctico)

En este primer anexo, se te propone el estudio de un programa en el cual podremos comprobar el funcionamiento de bastantes de las sentencias estudiadas.

PLANTEAMIENTO DEL PROGRAMA

Es frecuente desear que una información que se considera importante, se quiera mantener en secreto, a fin de que no esté al alcance de una persona no autorizada.

Para conseguir ello, conviene ENCRIPTAR *esta información. Es decir, cifrarla o criptografiarla. Más llanamente: ponerla en clave. De esta manera, a partir de una información inicial y aplicando la clave secreta que escojamos, obtendremos una información imposible de entender si no se dispone de la clave secreta y de la forma o proceso en que ésta se encriptó.*

Veamos un ejemplo fácil:

Tenemos el alfabeto natural, en mayúsculas:

A B C D E F G H I .....

y debajo lo ponemos al revés: (como clave)

Z Y W V U T S R Q .....

Está claro que la palabra BEBÉ *del primer alfabeto, se correspondería con la palabra* UYUY *del segundo alfabeto. En este caso la clave sería utilizar el alfabeto al revés, y el procedimiento sería emparejar un alfabeto con el otro.*

Cualquier aficionado a la CRIPTOGRAFIA, *o un espía principiante descubriría el sistema de cifrado que hemos empleado. Nosotros vamos a realizar un programa de encriptación mucho más complejo. Aunque podrías utilizar este programa como un juego, nuestra intención aquí es facilitarte un lugar de consulta de todo lo estudiado.*

FUNCIONAMIENTO DEL PROGRAMA

- *Nuestro programa aceptará un texto compuesto de letras y números, de una longitud máxima de* 200 *caracteres.*
- *La clave para encriptar nuestro mensaje la podemos introducir a nuestro gusto utilizando letras y números, con una longitud de* 0 a 62 *caracteres.*

  Si lo deseamos el propio MSX-C *se inventará una clave automática. Esto se determina entrando un* 1 *ó un* 2 *cuando el* MSX-C *nos lo pregunta.*
- *Una vez conseguido el mensaje encriptado tenemos dos opciones:*
  - *Pedir la* "desencriptación", *es decir, volver al mensaje inicial, para lo cual contestaremos* "S" *a la pregunta ¿Desea comprobar? del programa.*
  - *Ir a encriptar otro mensaje, para lo cual contestaremos con* N *ó* RETURN *a la misma pregunta.*
- *El programa, una vez cargado en memoria, se ejecuta con* F5.
- *Para finalizar la utilización del programa,* CTRL + STOP.

OPERATIVA INTERNA DEL PROGRAMA

Una vez entrada la clave y el texto, lo primero que se realiza es suprimir de la clave que hemos entrado (clave externa), las letras repetidas, obteniendo así una nueva clave que llamamos CLAVE COMPACTADA.

Ejemplo:

CLAVE EXTERNA: LA RANA MURIÓ

CLAVE COMPACTADA: LA RNMUIO

El espacio en blanco se considera un caracter como otro cualquiera, por esto el segundo espacio se suprime, porque está repetido.

A continuación, a un alfabeto completo que comprende las mayúsculas, las minúsculas y los números, el MSX-C *le quita o suprime los caracteres que coincidan con la clave compactada, y los añade al abecedario resultante, por su parte izquierda.*

Finalmente, al alfabeto resultante, lo partimos por la mitad y trasladamos su parte izquierda a su derecha.

Con ello hemos logrado una clave de trabajo que depende de la clave externa, que solo conoce el utilizador del programa.

CASO O EJEMPLO PRACTICO

Alfabeto original:

0123456789ABCDEFGHIJKLMNOPQRSTUWXYZabcdefghijkl mnopqrstuvwxyz.

Clave externa:

Perla del Cantábrico.

Clave compactada

Perla dCntbico

Alfabeto (clave) de encriptación

Paso 1° (letras de la clave compactada, se mueven a la izquierda del alfabeto).

Perla dCntbico0123456789ABDEFGHIJKLMNOQRSTUVWXYZ fghjkmpqsuvxyz.

Paso 2° (dividirlo e intercambiar sus partes).
JKLMNOQRSTUVWXYZfghjkmpqsuvwxyzPerla dCntbico0123456 789ABDEFGHI

Ya tenemos el alfabeto "desordenado", *con una ley que ha impuesto la clave externa.*

Si con este alfabeto encriptamos la frase:

HOY ES MARTES

quedará

h6HjbAj4d9BbA

Si no se conoce la clave, aunque se conozca el procedimiento, es difícil para un inexperto, entender lo que hay detrás de esta información.

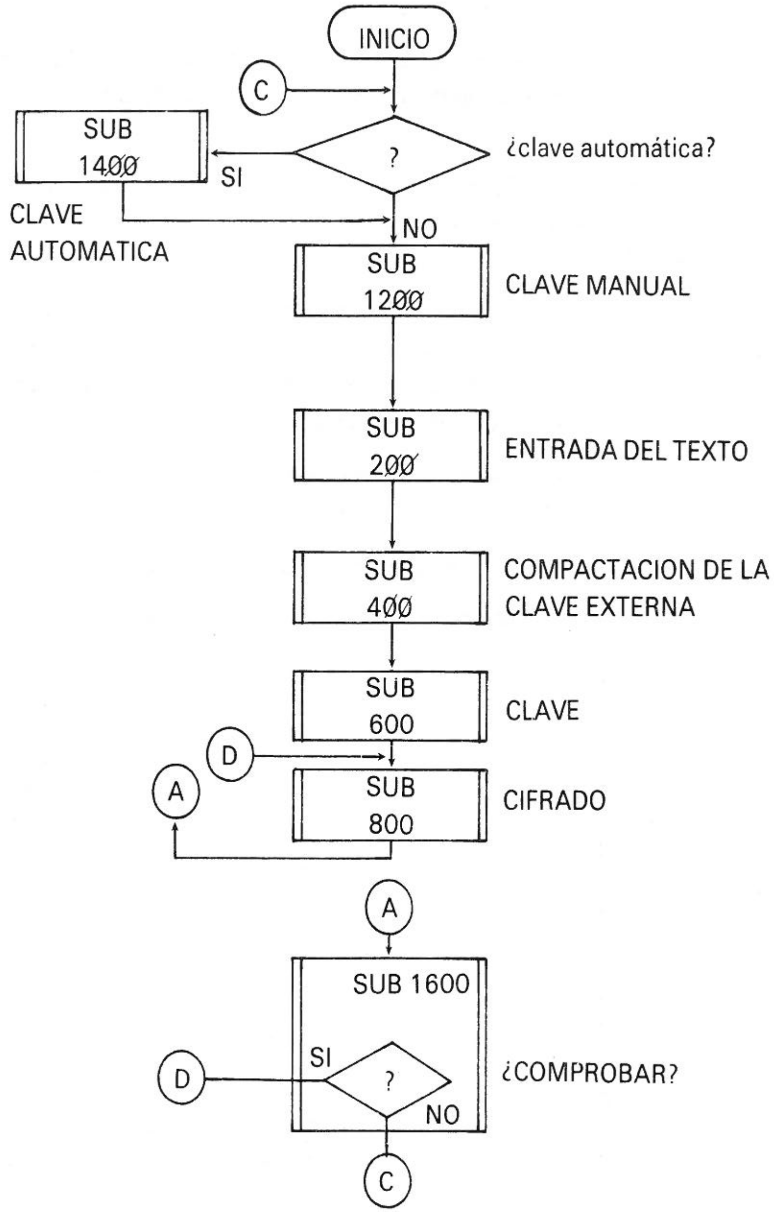
ANTES DE TECLEAR EL PROGRAMA

Se han puesto muchos textos y comentarios para que puedas comprender mejor la forma de trabajar de cada subrutina.

Puedes tomarlo como consulta si tienes dudas en la confección de tus futuros programas.

En el ANEXO-2° *tienes otros ejercicios variados para que puedas practicar y jugar con los mismos.*

ORDINOGRAMA DEL PROCESO



CODIFICACION DEL PROGRAMA

```
10 WIDTH 40: KEY OFF: CLS: CLEAR 10000
20 A$ = " 0123456789ABCDEFGHIJKLMNOPQRSTUVWX
   YZabcdefghijklmnopqrstuvwxyz"
30 GOSUB 1000 ' CABECERA
40 GOSUB 1080 ' MENU DE CLAVES
50 IF OP = 1 THEN GOSUB 1200 ELSE GOSUB 1400
60 GOSUB 200 ' ENTRADA DEL TEXTO
70 GOSUB 400 ' CLAVE SIN REDUNDANCIA
80 GOSUB 600 ' CLAVE INTERNA DE MANIOBRA
90 GOSUB 800 ' CIFRADO DEL TEXTO
100 GOSUB 1600
110 Q$ = LEFT$ (RQ$,1): IF Q$ = "S" OR Q$ = "s" THEN
    GOSUB 1800 ELSE GOTO 10
120 GOTO 90
200 ' ***************************
210 ' ENTRADA DEL TEXTO A CIFRAR
220 ' ***************************
230 CLS : PRINT "TEXTO A CIFRAR" : PRINT STRING$
    (14,"="): LINE INPUT TX$
240 L = CSRLIN
250 LG = LEN(TX$)
260 RETURN
400 '************************
410 'COMPACTACION DE LA CLAVE
420 '************************
430 CS$ = LEFT$(CE$,1)
440 B = LEN(CE$)
450 FOR A = 2 TO B
460 D = LEN(CS$)
470 CI$ = MID$(CE$,A,1)
480 FOR C = 1 TO D
490 IF CI$ = MID$(CS$,C,1) THEN 520
500 NEXT C
510 CS$ = CS$ + CI$
520 NEXT A
530 RETURN
600' **************************
610' CLAVE INTERNA DE MANIOBRA
620' **************************
630 B$ = A$
640 FOR A = 1 TO LEN (CS$)
650 PO = INSTR(B$,MID$(CS$,A,1))
660 B$ = LEFT$(B$,PO-1) + RIGHT$(B$,LEN(B$)-PO)
670 NEXT A
680 B$ = CS$ + B$ : B$ = RIGHT$(B$,31) + LEFT$
    (B$,32)
690 RETURN
800' *****************
810' CIFRADO DEL TEXTO
820' *****************
```

```
830 CR$ = ""
840 FOR A = 1 TO LG
850 PO = INSTR (A$,MID$(TX$,A,1))
860 CR$ = CR$ + MID$ (B$,PO,1)
870 NEXT A
880 RETURN
1000' ******************
1010' CABECERA DE MENUS
1020' ******************
1030 CLS
1040 LOCATE 7,0 : PRINT "PROGRAMA DE ENCRIPTOGRA-
     FIADO"
1050 LOCATE 7,1 : PRINT "= = = = = = = =  = =  = =
     = = = = = = = = = = = = = ="
1060 PRINT : PRINT
1070 RETURN
1080 LOCATE 11 : PRINT "opciones de clave' : LOCATE 11:
     PRINT "-------- -- -----"
1090 LOCATE 10, CSRLIN+4:PRINT"1 - CLAVE MANUAL":
     PRINT:PRINT: LOCATE 10: PRINT "2 - CLAVE AUTOMA-
     TICA"
1100 IF INKEY$ <> "" THEN 1100
1110 LOCATE 0,22: PRINT "Pulse 1 o 2";
1120 K$ = INKEY$
1130 IF K$ <> "1" AND K$ <> "2" THEN 1120
1140 IF K$ = "1" THEN OP=1 ELSE OP=2
```



```
1150 RETURN
1200' ******************************
1210' INTRODUCCION DE CLAVE MANUAL
1220' ****************************
1230 GOSUB 1000
1240 CE$ = ""
1250 LOCATE 14,4 : PRINT "CLAVE MANUAL": PRINT TAB(14)
     "----- ------"
1260 LOCATE 0,12: LINE INPUT "CLAVE:";CE$
1270 RETURN
1400' ********************************
1410' COMPOSICION DE CLAVE AUTOMATICA
1420' ********************************
1430 GOSUB 1000
1440 CE$ = ""
1450 LOCATE 12,4 : PRINT "CLAVE AUTOMATICA": PRINT
     TAB(12) "----- ----------"
1460 LOCATE 0,6: INPUT "cuantos caracteres tendra la clave";
     CH
1470 IF CH<1 OR CH>63 THEN 1400
1480 Y = RND (-TIME)
1490 FOR A = 1 TO CH
1500 X = INT (65+58*RND(45))
1510 IF X>90 AND X<97 THEN 1500
1520 CE$ = CE$ + CHR$ (X)
1530 NEXT A
```

```
1540 LOCATE 0,12: PRINT "Clave:";CE$
1550 LOCATE 0,22 : PRINT "Pulsar cualquier tecla blanca"
1560 IF INKEY$ <> "" THEN 1560
1570 K$ = INKEY$
1580 IF K$ = "" THEN 1570
1590 RETURN
1600' **********************************
1610 REM     IMPRESION DEL TEXTO CIFRADO
1620'***********************************
1630 LOCATE, L + 1: PRINT "CLAVE EXTERNA": PRINT
     "= = = = =  = = = = = = ="
1640 PRINT CE$
1650 PRINT: PRINT "CLAVE COMPACTADA"
1660 PRINT "= = = = =  = = = = = = = = = ="
1670 PRINT CS$
1680 PRINT: PRINT "ALFABETO DE ENCRIPTACION"
1690 PRINT "= = = = = = = =   = =   = = = = ="
            = = = = = = ="
1700 PRINT B$
1710 PRINT: PRINT "TEXTO ENCRIPTADO"
1720 PRINT "= = = = =  = = = = = = = = = ="
1730 PRINT CR$
1740 LOCATE 0,23: PRINT "Desea comprobar (S,# S)";:RQ$ =
     INPUT$(1)
1750 RETURN
1800' *************************
```



```
1810' COMPROBACION DEL CIFRADO
1820' **************************
1830 GOSUB 1000
1840 SWAP A$, B$: SWAP CR$, TX$
1850 RETURN
```

ANEXO 2º (Ejercicios prácticos)

En las páginas siguientes encontrarás una serie de ejercicios de perfeccionamiento. En todos ellos hallarás primero un enunciado y en la siguiente página una solución al mismo. Trata por ti mismo de encontrar una solución antes de consultar la que aquí te brindamos.

En este, nuestro primer contacto contigo, este es el momento de la despedida.

Esperamos que entre tú y el MSX-C BASIC *haya nacido una incipiente amistad que, con tu constancia, llegue a ser de gran profundidad.*

¡¡ Hasta muy pronto, en nuestro segundo volumen !!

PRACTICAS MSX-BASIC - 1

ENUNCIADO:

Calcular la longitud de una circunferencia entrando por teclado simplemente el radio. (valor de Pi = 3,1416)

El alumno entrará el siguiente programa y calculará las longitudes para radios de: 1 mt., 7 mts. y 237 mts.

PRACTICAS MSX-BASIC - 1

SOLUCION:

PROGRAMA 1

```
10 INPUT R
20        L = 2*3.1416*R
30 PRINT L
```

RESULTADOS CONSEGUIDOS: R = 1, LONGITUD = 6,2832
R = 7, LONGITUD = 43,9824
R = 237, LONGITUD = 1489,1184

PRACTICAS MSX-BASIC - 2

ENUNCIADO:

En las siguientes instrucciones hay errores de sintaxis. Entrarlas al MSX-C y corregir los errores.

```
10 B = 7
20 A = 10
30 A x B = C
40 5 = D
50 C/D = R
60 13 = N
70 R - N = B
80 PRINT B
90 END
```

SOLUCION:

PROGRAMA 2

10 B = 7	B = 7
20 A = 10	A = 10
30 A x B = C	C = A * B
40 5 = D	D = 5
50 C/D = R	R = C/D
60 13 = N	N = 13
70 R - N = B	B = R - N
80 PRINT B	PRINT B
90 END	END

El valor final de "B", es = 1

ENUNCIADO

Corrección de errores.

Queremos ejecutar la operación: $C = \frac{118 \times 2{,}6}{5 \times 7{,}3}$

El resultado correcto debe ser: C = 8,405479

Indicar porque son erróneos los siguientes planteamientos y escribir el programa correcto.

PROGRAMA 1:

```
10 118 x 2,6/5 x 7,3 = C
20 PRINT C
30 END
```

PROGRAMA 2:

```
40 C = 118 * 2,6/5 * 7,3
50 PRINT C
60 END
```

SOLUCION:

PROGRAMA 1:

```
10 118 x 2,6/5 x 7,3 = C
20 PRINT C
30 END                    (ERROR DE SINTAXIS)
```

PROGRAMA 2:

```
40 C = 118 * 2.6/5 * 7.3
50 PRINT C
60 END                    (ERROR DE LOGICA)
```

PROGRAMA CORRECTO:

```
10 C = (118 * 2.6) / (5 * 7.3)
20 PRINT C
30 END
```


ENUNCIADO

Queremos entrar e imprimir la fecha de hoy. Si el programa a continuación no lo permite, modificarlo:

PROGRAMA:

```
10 INPUT F        (Teclear "XX/XX/XX")
20 PRINT F
30 END
```

SOLUCION:

```
10 INPUT F$          (Teclear "XX/XX/XX")
20 PRINT F$
30 END
```


ENUNCIADO:

Realizar un programa que calcule el valor medio de cuatro números que se introduzcan por teclado.

SOLUCION:

EJEMPLO:

```
100 INPUT A, B, C, D
110 X = (A + B + C + D) / 4
120 PRINT X
130 END
```


ENUNCIADO:

Realizar un programa que convierta grados Fahrenheit a grados Celsius.

La relación es: C = (5/9) x (F-32)

Dar como resultados:

F 212 C = 100

F 183 C = 83,88889

F 48 C = 8,88889

SOLUCION:

```
100 PRINT "TECLEAR N. DE GRADOS F";
110 INPUT F
120 C = (5/9)*(F-32)
130 PRINT F; "GRADOS F SON" ; C; "GRADOS C"
140 END
```


ENUNCIADO:

Realizar un programa que calcule e imprima la suma y el producto de dos números. Una vez calculado ello, e imprimidos ambos valores, el programa dejará un espacio en blanco y se posicionará automaticamente para entrar una nueva pareja de valores.

PRACTICAS MSX-BASIC - 7

SOLUCION:

```
100 PRINT "ENTRAR A";
110 INPUT A
120 PRINT "ENTRAR B";
130 INPUT B
140 PRINT "SUMA DE A + B ES"; A + B
150 PRINT "PRODUCTO DE AxB ES"; (A * B)
160 PRINT
170 PRINT
180 GO TO 100
190 END
```


PRACTICAS MSX-BASIC - 8

ENUNCIADO:

Realizar un programa que calcule e imprima todos los números pares comprendidos entre el 1 el 100, ambos inclusive.

PRACTICAS MSX-BASIC - 8

SOLUCION:

```
10 X = 2
20 PRINT X
30 PRINT
40 X = X + 2
50 IF X < 101 THEN 20
60 END
```


PRACTICAS MSX-BASIC - 9

ENUNCIADO:

Utilización de "Read" y "Data".

Imaginemos que tenemos una serie de números en la sentencia "DATA", los cuales queremos promediar. No sabemos anticipadamente los números que hay, pero el último será siempre el 9999, el cual no debe tomarse en cuenta para el promedio.

El programa siguiente resuelve este problema, pero contiene errores en las líneas 100, 110, 160 y 170. Estudiar la lógica del programa y modificar líneas indicadas.

PROGRAMA:

```
100 S = 1
110 N = 1
120 READ X
130 IF X = 9999 THEN 170
140 S = S + X
150 N = N + 1     1
160 GOTO 100
170 A = S * N
180 PRINT A
190 DATA 4, 2, 3, 6, 5, 9999
200 END
```

PRACTICAS MSX-BASIC - 9

SOLUCION

PROGRAMA:

```
100 S = 1
110 N = 1
120 READ X
130 IF X = 9999 THEN 170
140 S = S + X
150 N = N + 1
160 GOTO 100
170 A = S*N
180 PRINT A
190 DATA 4, 2, 3, 6, 5, 9999
200 END
```

CORRECCION:

```
100 S = 0
110 N = 0
160 GOTO 120
170 A = S/N
```

SOLUCION = 4

PRACTICAS MSX-BASIC - 10

SENTENCIAS FOR y NEXT

ENUNCIADO:

De parecida forma a otro ejercicio anterior, queremos calcular el valor medio (promedio) de una serie de números que tecleemos al sistema.

El programa preguntará:

¿Cuantos números se teclearán?

Luego permitirá la entrada de los números, uno a uno.

Finalmente se calculará el valor medio y se imprimirá.

Utilizar las sentencias "FOR y "NEXT"

SOLUCION:

```
100  PRINT "CUANTOS NUMEROS SE TECLEARAN?";
110  INPUT N
120  PRINT "ENTRE LOS NUMEROS, UNO A UNO"
130  S = 0
140  FOR I = 1 TO N
150  INPUT X
160  S = S + X
170  NEXT I
180  A = S/N
190  PRINT "EL VALOR MEDIO ES"; A : GOTO 100
```


ENUNCIADO:

Vamos a realizar un programa que permita la entrada de una cadena de caracteres y la impima en orden inverso al entrado.

SOLUCION:

```
100  INPUT A$
110  FOR X = LEN (A$) TO 1 STEP-1
120  PRINT MID$ (A$, X, 1);
130  NEXT X
140  PRINT: GOTO 1ØØ
```


ENUNCIADO:

Si A$ = "Que dia viene Laura", escribir una función que extraiga "Dia viene".

SOLUCION:

```
100 A$ = "QUE DIA VIENE LAURA"
110 B$ = MID$(A$,5,9)
120 PRINT B$
```


ENUNCIADO:

¿Que saldrá impreso si ejecutamos el siguiente programa?

```
100 FOR N = 65 TO 80
110 FOR M = 65 TO N
120 PRINT CHR$(M);
130 NEXT M
140 PRINT
150 NEXT N
160 END
```

SOLUCION:

A

AB

ABC

ABCD.......P

ENUNCIADO:

¿Que saldrá impreso si ejecutamos el siguiente programa?

(Piénsalo y adivínalo antes de hacerlo)

```
100 A = 1
110 GOSUB 200
120 A = A + 4
130 GOSUB 200
140 A = A - 2
150 GOSUB 200
160 END
200 REM SUBRUTINA
210 IF A < 2 THEN 250
220 IF A = 3 THEN 270
230 PRINT "ROJO"
240 GOTO 280
250 PRINT "BLANCO"
260 GOTO 280
270 PRINT "AZUL"
280 RETURN
```

PRACTICAS MSX-BASIC - 14

SOLUCION:

BLANCO

ROJO

AZUL

INTRODUCCION A LA IMPRESORA PLOTTER EN COLORES, SONY PRN-C41

A - INTRODUCCION

La acelerada evolución de la microinformática en los últimos cinco años, nos ha permitido ser espectadores de la masificación en el uso de productos, que poco tiempo atrás estaba reservado a un porcentaje relativamente pequeño de especialistas en el tema informático.

El continuo abaratamiento en la producción de componentes electrónicos, nos permite poner a su alcance productos como el HIT BIT *de* SONY *que, dentro de la llamada "gama de computadores domésticos", ofrecen posibilidades de proceso que exceden en mucho el entorno familiar, aportando soluciones al mercado profesional e incluso empresarial.*

En los capítulos anteriores se han transmitido los conocimientos básicos del lenguaje BASIC-MSX *para iniciarle en la utilización rentable de su* HIT BIT.

SONY *ha sido consciente de que la inquietud de nuevos conocimientos por parte del alumno que ha llegado al final del presente libro, le llevarán a querer disponer de una serie de elementos adicionales que le permitan:*

- *Almacenar programas y datos a manejar por los mismos.*
- *Obtener listados impresos tanto de programas como de resultados*
- *Manejar el mundo de los gráficos y dibujos en general.*
- *Enriquecer los resultados anteriores con diferentes colores de representación.*
- *Etc.*

Para todo ello SONY *pone a disposición de los usuarios de su* HIT BIT, *una amplia gama de periféricos conectables al mismo. De entre ellos, y por las características diferenciales frente a la oferta actual del mercado, dedicamos estas páginas a mostrarle algunas de las características de la impresora plotter* SONY PRN-C41.

B.- **PRN-C41** (Características fundamentales)

1.- OBJETIVO

La PRN-C41 *permite ser utilizado como una impresora convencional del alfabeto arábigo o como un instrumento de ploterización, pudiendo mezclar ambas técnicas de impresión y dibujo en cualquier punto del impreso, sin ningún paso o transición intermedios.*

Si bien ambas funciones están disponibles en otros periféricos del mercado, el reducido precio de la PRN-C41 *y su conectabilidad al* HIT BIT *de* SONY *justifican el éxito de ventas obtenido.*

2.- ESPECIFICACIONES

Conjuntamente con la impresora - plotter se suministra al comprador un manual en castellano con información más exhaustiva del producto.
Destacamos aquí solo algunas de las especificaciones principales.

FUNCIONES DE IMPRESION	
COLORES	: CUATRO (NEGRO, AZUL, VERDE y ROJO)
Sistema	: GIRATORIO, CON CUATRO BOLIGRAFOS
Tipos de caracteres	: 252
Tamaños de caracter	: HASTA 16 DISTINTOS
Caracteres por línea	: EN PAPEL A4, 160 DE TAMAÑO MINIMO
Velocidad	: MAXIMA DE 10 CARACTERES/SEGUNDO

Velocidad de impresión gráfica : X, Y: 57 mm/seg.
: dirección de 45°: 81 mm/seg.

Area de impresión gráfica:

Tamaño del papel	100 mm	114 mm	A5	B5	A4
x-dirección	410	480	650	820	960
	(82)	(96)	(130)	(164)	(192)
+ y-dirección	30	--	30	30	30
	(6)		(6)	(6)	(6)
-y-dirección	601	--	919	1149	1354
	(120,2)		(183,8)	(229,8)	(270,8)

Unidad: paso (entre paréntesis: mm)

TIPO DE PAPEL	
Cuartillas sueltas:	ANCHURA 100 a 216 mm. TAMAÑO MAXIMO: DIN A4
Rollo de papel contínuo:	SE SUMINISTRA COMO ACCESORIO ESTANDARD. DIAMETRO EXTERIOR: < 70 mm. DIAMETRO INTERIOR: < 12 mm. ANCHURA: 114 mm.

3.- FUNCIONES POR TECLA MANUAL

Una serie de acciones y funciones están al alcance del usuario mediante la depresión de las teclas incorporadas a la PRN-C41. *Las más relevantes son:*

CHANGE COLOR (Cambio de color)

Gira el portabolígrafos al color elegido.

△

Produce el avance del papel.

▽

Produce el retroceso del papel

RESET (Restaurar)

Repone el impresor-trazador cuando se haya interrumpido la impresión.

CHANGE COLOR + △

Se imprimen automaticamente los patrones siguientes:

□ (negro) □ (azul) □ (verde) □ (rojo)

CHANGE COLOR + ▽

Se imprimen automaticamente todos los caracteres. Cada 16 caracteres, cambia automaticamente el color de la impresión.

C.- UTILIZACION BASICA

1.- UTILIZACION BASICA

Se utiliza el comando LLIST *con funciones similares en la impresora a las del comando* LIST *en pantalla.*

Para imprimir caracteres de programas del BASIC-MSX *se emplea el comando* LPRINT, *con funciones similares en el impresor a las del comando* PRINT *en pantalla.*

2.- UTILIZACION AVANZADA

MODO DE TEXTO

En este modo de texto se utiliza el mandato LLIST *del* BASIC-MSX *para imprimir listas de programas del* BASIC-MSX, *y el mandato* LPRINT *para imprimir caracteres.*

En este modo, la impresora plotter podrá controlarse enviándole códigos desde el ordenador.

A continuación se ofrecen los códigos y sus funciones.

CODIGOS

NOMBRE	CODIGO	FUNCIONES
Retorno del carro	0DH	*Cuando el interruptor* DIP 4 *de la parte posterior de la impresora esté en* ON: *la posición del bolígrafo se moverá hasta el comienzo de la línea siguiente* OFF: *la posición del bolígrafo se moverá hasta el comienzo de la línea.*
Avance de línea	0AH	*Avanza una línea*
Retroceso	08H	*La posición del bolígrafo retrocede el espacio de carácter. (Cuando el bolígrafo se encuentre al comienzo de la línea, este código se ignorará).*

NOMBRE	CODIGO	FUNCIONES
Retroceso de línea	0BH	*Hace retroceder el papel hasta la línea anterior.*
Modo de gráficos	1BH + "•"	*Establece el impresor/trazador en el modo de gráficos.*
Establecimiento de escala	12H + "0"-"15"	*Establece el tamaño de los caracteres impresos. "0" indica el tamaño más pequeño.*
Establecimiento de color	1BH + "C" + "0"-"3"	*Establece el color del bolígrafo 0: negro 1: azul 2: verde 3: rojo*
Tope de formulario	0CH	*Avanza el papel hasta una posición situada a 297 mm. de la posición de comienzo de la impresión.*

MODO DE GRAFICOS

Enviando mandatos específicos, se pueden imprimir gráficos, cambiar los colores, imprimir caracteres, etc.

MANDATOS

NOMBRE	MANDATO	FUNCIONES	EJEMPLO
Inicialización	I	*Establece la posición actual del bolígrafo como origen (intersección de los ejes x e y).*	
Origen	H	*Mueve el bolígrafo al origen sin trazar nada.*	
Inicialización total	A	*Mueve el bolígrafo hasta la posición extrema izquierda, establece esta posición como origen, y pone en la impresora en el modo de texto.*	

NOMBRE	MANDATO	FUNCIONES	EJEMPLO
Nueva línea	F	*Mueve el bolígrafo hasta el comienzo de la línea siguiente.*	
Cambio de color	Cn (n = 0-3)	*Establece el color 0: negro, 1: azul 2: verde 3: rojo*	*C0*
Establecim. de escala	Sn (n = 0-15)	*Establece el tamaño de los caracteres impresos*	*S5*
Dirección de impresión	Qn (n = 0-3)	*Establece la dirección de los caracteres impresos* (0 ↑, 3 ←, → 1, ↓ 2)	*Q3*
Impresión	Pchrs.	*Imprime caracteres*	*PABC*
Tipo de línea	Ln (n = 0-15)	*Selecciona el trazado de línea contínua o discontínua.* *n = 0,15 línea contínua* *n = 1-14 línea discontínua (Cuanto mayor sea el número especificado, mayor será la discontinuidad de la línea).*	*L0* *L10*
Trazado	Dx1,y1,x2,y2... (-999≤x,y≤999)	*Traza una línea desde la posición actual del bolígrafo hasta (x1,y1), después desde (x1,y1) hasta (x2,y2)...*	*D0,0,0,5, 5,5,5,10*
Trazado relativo	J△x1,△y1, △x2,△y2,... (-999≤△x,△y ≤999)	*Traza una línea desde la posición actual del bolígrafo hasta △x1 en dirección horizontal y △y1 en dirección vertical, después tal punto hasta △x2 en dirección horizontal y △y2 en dirección vertical, etc.*	*J0,5,5,0 0,5,5,0*

NOMBRE	MANDATO	FUNCIONES	EJEMPLO
Movimiento	Mx,y (-999≤x,y≤999)	*Mueve el bolígrafo hasta (x,y).*	*M10,20*
Movimiento relativo	R△x,△y (-999≤△x,△y 999)	*Mueve el bolígrafo△x en dirección horizontal y △ y en dirección vertical.*	*R15,15*

D.- EJEMPLOS

A continuación se incluyen algunos ejemplos y comentarios.

Para cada ejemplo se detalla el programa utilizado y el resultado obtenido con su ejecución.

Tanto el listado del programa como el dibujo que le sigue se ha obtenido con la PRN-C41.

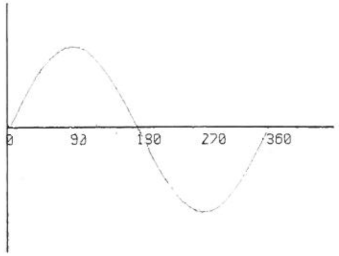
Cabe destacar que para facilitar la impresión de ilustraciones o caracteres especiales, existen en el mercado cartuchos con software apropiado para confeccionar tarjetas de navidad, de bautizo, rótulos, etc.

1.- COMO TRAZAR UNA CURVA SENOIDAL EN EL IMPRESOR-TRAZADOR

Programa	Comentario
1Ø LPRINT:LPRINT CHR$(&H1B) + "#"	*Envío de un código de retorno del carro y establecimiento de la impresora en el modo de gráficos.*
2Ø LPRINT "S1"	*Establecimiento del tamaño de los caracteres a "1"*
3Ø LPRINT "C2"	*Selección del bolígrafo verde*
4Ø LPRINT "P*** SIN CURVE ***"	*Impresión de "*** SIN CURVE ***"*
5Ø LPRINT "F"	*Movimiento del bolígrafo hasta el comienzo de la línea siguiente.*
6Ø LPRINT "CØ"	*Selección del bolígrafo negro*
7Ø LPRINT "R1Ø,Ø"	*Movimiento del bolígrafo 10 pasos en dirección horizontal.*
8Ø LPRINT "JØ,-3ØØ"	*Trazado de una línea vertical de -300 pasos a partir de la posición del bolígrafo.*
9Ø LPRINT "RØ,15Ø"	*Movimiento del bolígrafo 150 pasos en dirección vertical.*
1ØØ LPRINT "I"	*Reposición allí del origen.*
11Ø LPRINT "J45Ø,Ø"	*Trazado de una línea horizontal de 450 pasos a partir del origen.*
12Ø LPRINT "H"	*Movimiento del bolígrafo hasta el origen.*
13Ø LPRINT "C3"	*Selección del bolígrafo rojo*
14Ø FOR J = Ø TO 36Ø STEP 9Ø 15Ø LPRINT "M";J-15;",";-2Ø 16Ø LPRINT "P";J 17Ø NEXT	*Marcación de la graduación sobre el eje x*
18Ø LPRINT "H"	*Movimiento del bolígrafo hasta el origen*
19Ø LPRINT "C1"	*Selección del bolígrafo azul*
2ØØ FOR X = Ø TO 36Ø STEP 1Ø 21Ø I = X*3. 14/18Ø 22Ø Y = SIN(I)*1ØØ 23Ø LPRINT "D";X;",";Y 24Ø NEXT X	*Trazado de una curva senoidal*
25Ø LPRINT "H"	*Movimiento del bolígrafo hasta el origen*
26Ø LPRINT "A"	*Establecimiento de la impresora en el modo de texto*
27Ø END	

RESULTADO DE LA EJECUCION

*** SIN CURVE ***



(Impresión realizada por sistema convencional de imprenta gráfica, sin colores).

2.- VARIACION AL EJEMPLO ANTERIOR

Basado en el ejemplo anterior, se mantiene el trazado de la curva senoidal, añadiendose a su vez una cosenoidal.

Para ello, y comparando el programa anterior con el de este ejemplo, observamos que se han llevado a término las siguientes modificaciones:

- *Se ha anulado la sentencia 270*
- *Se han añadido las sentencias 200, 210, 240, 270, 290, 300 y 310.*
- *Se ha modificado la sentencia 40.*

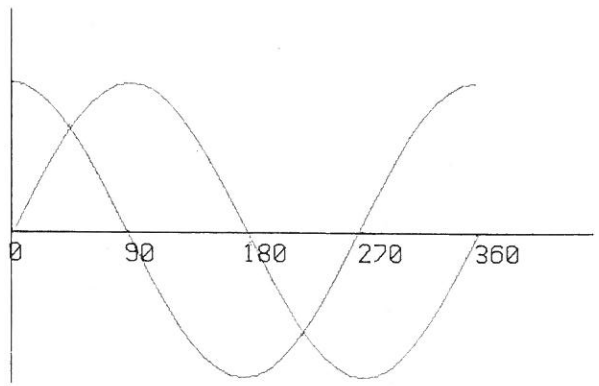
El listado del programa y el dibujo a continuación, han sido dibujados por el impresor-trazador.

– LISTADO DEL PROGRAMA UTILIZADO

```
10 LPRINT:LPRINT CHR$(&H1B)+"#"
20 LPRINT "S1"
30 LPRINT "C2"
40 LPRINT "P*** curva senoidal y co-senoidal ***"
50 LPRINT "F"
60 LPRINT "C0"
70 LPRINT "R10,0"
80 LPRINT "J0,-300"
90 LPRINT "R0,150"
100 LPRINT "I"
110 LPRINT "J450,0"
120 LPRINT "H"
130 LPRINT "C3"
140 FOR J=0 TO 360 STEP 90
150 LPRINT "M";J-15;",";-20
160 LPRINT "P";J
170 NEXT
180 LPRINT "H"
190 LPRINT "C1"
200 FOR A=1 TO 2
210 LPRINT "H"
220 FOR X=0 TO 360 STEP 10
230 I=X*3.14/180
240 ON A GOSUB 280,300
250 LPRINT "D";X;",";Y
260 NEXT X
270 NEXT A
280 Y=SIN(I)*100
290 RETURN
300 Y=COS(I)*100
310 RETURN
360 LPRINT "H"
370 END
```

RESULTADO DE LA EJECUCION

*** curva senoidal y co senoidal ***



3.- EJEMPLO DE LINEAS ENVOLVENTES

– Listado del programa utilizado.

```
10 LPRINT CHR$(&H1B)+"#"
20 LPRINT "M320,-448"
30 LPRINT "I"
40 LPRINT "C3"
50 FOR N=1 TO 16
60 XO=N*20
70 YO=0
80 LPRINT "M";XO;",";YO
90 X=320-N*20
100 Y=-N*40
110 LPRINT "J";X;",";Y
120 NEXT N
130 FOR N=1 TO 16
140 XO=320
150 YO=-N*40
160 LPRINT "M";XO;",";YO
170 LPRINT "J";-N*20;",";-640+N*40
180 NEXT N
190 FOR N=1 TO 16
200 XO=320-N*20
210 YO=-640
220 LPRINT "M";XO;",";YO
230 LPRINT "J";-320+N*20;",";N*40
240 NEXT N
250 FOR N=1 TO 16
260 XO=0
270 YO=-640+N*40
280 LPRINT "M";XO;",";YO
290 LPRINT "J";N*20;",";640-N*40
300 NEXT N
310 LPRINT "M25,-700"
320 LPRINT "S3"
330 LPRINT "C2"
340 LPRINT "PEnvolventes"
```


RESULTADO DE LA EJECUCION

